Nepali Speech Recognition

Aavaas Gajurel (068/BCT/501)
Anup Pokhrel (068/BCT/505)
Manish K. Sharma (068/BCT/523)

A Final Year Project Report submitted

to the

Department of Electronics and Computer Engineering
Central Campus
Tribhuvan University

Supervisor:Dr. Basanta Joshi

Bhadra 28 ,2072

LETTER OF APPROVAL

The undersigned certify that they have read, and recommended to the Institute of Engineering for acceptance, a project report entitled "Nepali Speech Recognition" submitted by Aavaas Gajurel(068/BCT/501), Anup Pokhrel(068/BCT/505) and Manish Kumar Sharma(068/BCT/523) in partial fulfilment of the requirements for the Bachelor's degree in Computer Engineering.

DATE OF APPROVAL: 29.05.2072

COPYRIGHT

ACKNOWLEDGEMENT

This project is done in partial fulfillment of the requirement for the Bachelors in Computer Engineering.

First of all, we would like to thank our Department of Electronics and Computer Engineering for providing us the opportunity to learn and implement our knowledge in the form of major project. With great pleasure, we would like to acknowledge the contributions of various individuals for our project, “Nepali Speech Recognition”.

We owe a debt of gratitude towards our project supervisor, Dr. Basanta Joshi, for his proper guidance, encouragement and support throughout the project development. We would also like to thank Dr. Dibakar Raj Pant, Head of Department, Dr. Nanda Bikram Adhikari and Mr. Dinesh Baniya Kshetri, Deputy Head of Department of DOECE for timely guidance.

We would also like to extend our gratitude to Assoc. Prof. Tika Upreti, the internal examiner and Ms. Pratima Pradhan, the external examiner of the project. We would also like to acknowledge the authors of various papers and developers of the programming libraries that we have referenced and utilised in our project. We respect all the person hours gone into research and writing of such articles. This project wouldn’t have been possible without them being kind enough to share their knowledge.

We would also like to acknowledge our teachers and friends for their motivation and support for our project. Finally, we would like to thank all the people who are directly or indirectly related during our study and preparation of this project.

Aavaas Gajurel (068/BCT/501)
Anup Pokhrel (068/BCT/505)
Manish K. Sharma (068/BCT/523)

ABSTRACT

It has been observed that there are various cases where computers cannot simply be used due to non-feasible method of input or due to lack of knowledge of foreign languages used in a major portion of Nepalese population. The other problem is difficulty in typing in Nepali language font such as 'Preeti '. Thus the proposed system is a Speech to Text converter for Nepali language that could provide a base for solution of the problem.

This report provides an insight into various methods and techniques used and to be used to implement the stated system. The feature extraction is carried out via MFCC along with noise reduction via Spectral Subtraction. The ngram based language model trained with data from various news sources, can further enhance the accuracy of the system. Methods such as HMMs are used in speech recognition because a speech signal can be viewed as a piecewise stationary signal or a short-time stationary signal. This algorithm allows for facility of continuous speech recognition and conversion. HMM method is simple and computationally feasible.

The final output is a system capable of taking voice inputs for a specific domain and provide output as a series of words based on the language model stored with an easy to use GUI.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

1. ASR : Automatic Speech Recognition
2. VAD : Voice Activity Detection
3. HMM : Hidden Markov Model
4. ZCR : Zero Crossing Rate
5. SPL : Sound Pressure Level
6. STE : Short Term Energy
7. MFCC : Mel Frequency Cepstrum Coefficients
8. GMM : Gaussian Mixture Model
9. SNR : Signal to Noise Ratio

1. INTRODUCTION

1.1. Background

Software has become an essential part of human life. It can be implemented in every possible work possibility scenarios. The complexity of the work which was to be handled by human is accomplished by the use of an automated system in many forms. This has eased operation of various corporation as well as individuals. From its dawn since late 1980s where it was used for computational tasks, word processing and spreadsheet, computer technology growth and application hasn't looked back. Today powerful applications can be run from a single computer at an individual's residence.

However, Nepal is a country with about 66 percent average literacy rate, as of 2011. The percentage of people having acquaintance with computers and related technology is much less than the above figure. Thus, Nepal is yet to receive such widespread use of computer technology to facilitate everyday task. Even though many projects are being developed by the government, there has not been considerable advancement in getting the rural population to use the computer in significant amount.

The inputs to a computer system is provided in general by the use of keyboard, mouse, stylus or touch. Recently, a rapid growth in wearable technology has introduced speech as a candidate alternative to afore mentioned input methods. However, these systems are limited to languages not spoken by Nepalese in general.

Thus, the system, that has been designed, can lay basis for computer operation by speech in Nepali Language. Speech Recognition system is a system that is responsible for conversion of spoken audio input into a text providing meaningful output. This system is made up of language models, prediction algorithms and digital signal process algorithms. The output then could be used for a number of tasks ranging from simple text conversion upto operation of a complete software package to the operating system itself.

There have already been numerous implementation of Speech To Text Algorithms for various languages in numerous domains and discovering widespread usage. It is to be noted that there have also been some attempts at Nepali Speech Recognition system development. But, no significant traces of such implementations have been found. This project aims to further contribute to the field. The project also aims to improve the accuracy of the system using modern hardware and algorithm optimizations.

The proposed system could have applications in health care, telecommunications, and support centers for automated query processing. It could also be used by people lacking appropriate education to operate computers and by disabled people.

1.2. Motivation

It has been observed that there are various cases where computers cannot simply be used due to non-feasible method of input or due to lack of knowledge of foreign languages used in a major portion of Nepalese population. The most optimum way to enhance computer system usage is to develop a Nepali Speech Processing System via a text to speech conversion system.

Following are major application areas that have motivated us to build the proposed system, which might provide foundation for such usage:

1. Tedious and error prone Nepali text typing.
2. Failure from side of non-speech impaired Nepali speaking people to operate computer.
3. Fundamental usage of computer (for communicating) by Nepali speaking persons with little or no knowledge of computers.

Furthermore, on reaching an acceptable level of accuracy the system could find usage in healthcare, telecommunication, etc. Such widespread potential use is the major motivation factor in this project.

1.3. Objectives

The basic objective of this project is to design a system that solves the problem of Speech to text conversion for Nepali language. Moreover, this project is intended to allow us to learn about various aspects of Artificial Intelligence, Data Management and various other technologies.

1.3.1. Project Objectives

The basic objectives of our project are as follows:

1. To allow users to provide input via speech in Nepali Language using easy to use User Interface (UI).
2. To design a crawler capable of routine updating the knowledgebase.
3. To make use of domain specific knowledge for text prediction for the domain.
4. Provide the output Nepali text with improved accuracy.

1.3.2. Academic Objectives

In terms of academic level this project helps us learn more on various terminologies and development strategies in sectors such as, Natural Language Processing, Digital Signal Processing, etc. Because of this project work we should be able to learn more about scripting language like Python. This project also helps us in knowing about various data passing and parsing techniques. Besides these, we will obtain a new level of academic step learning about data formats, conversion techiques,etc. In brief the following are major academic objective of this report concerning the system has the following credentials:-

1. To help us get an insight into concepts of Language and its Models.
2. To understand about training and classification techniques.
3. To understand the basics of speech processing
4. To help us analyze the various types of requirements.
5. To learn about techniques of data passing among various modules.

1.4. Scope

The scope of the project reaches vast arenas as speech is the fundamental ways by which the humans communicate. The maximum product could be a human like understanding of the language in terms of speech but that being unrealistic in for the time and effort possible for the major project, we have limited our scope to some achievable domain.

We have built a product that will be capable of understanding the Nepali speech in a limited vocabulary domain. As we need training corpus for our text and the availability of formal news material is effortless, we will initially restrict our domain to formal texts. Thus the expected outcome of our major project is being able to understand natural human speech in Nepali and display that in written Unicode string of Nepali characters being restricted to News domain.

The system will be built for the desktop computer and will not be made for hand held devices. We plan to use open source methodologies and tools thus to make the system cross platform such that it can run on any operating system (mainly Windows, Linux and MacOS). We will use the noise reduction techniques but we will require a separate microphone for the optimal sound capture. Thus this project is not aiming for far field speech recognition rather will use recorded voice at close proximity.

The processing of all the implemented algorithms will be a heavy task and thus will require ample time. Thus, in moderate hardware, we expect some delay from recording of the speech signal to the text output. That is to mean, in the first phase without the optimizations, we are not aiming for real-time speech recognition. But with further optimization on algorithm, threaded implementation and use of parallel hardware, we should be able to do real-time transcribing.

But our project has huge future scope and potential. We can improve our program to be more accurate in understanding complex speech patterns and also generalize the domain of understanding in the future. Speech Recognition if made satisfactorily accurate, can be used in wide applications. We list some of them below.

- Use as an input methodology to input text to the system: -This will be helpful for situations where input via hardware is inadequate i.e smartphones, public service stations, etc -This will also be useful for people with disabilities for conventional input methods.

- Used by military for automatic wiretapping for criminal detection and threat monitoring -This cannot be done manually due to the nature of the problem and the vastness of the data. But if given enough resources can be automated. -System can be made such that privacy is not invaded, which would be hard to recreate by using personnel's.
- Use in guidance systems in public places: -Public information systems are best for implementing speech recognition techniques for quick and to the point queries. -if natural language understanding is implemented, it will be best for general public.
- Automation of call-center assistance: -Currently, all mundane tasks in call-center requires human to understand and correspond. If the system is made intelligent, the human part can be completely replaced.
- Automatic transcribing of court cases and similar scenarios: -This system will be useful in high frequency and un-interruptible situations and initially can be used for proof reading of the human transcriber.
- Speech controlled computer user interface. Etc: -The use of speech for computer interaction will surely be the norm in the future, and thus having a Nepali speech recognition system will certainly provide us with an edge.

1.5. Requirements

1.5.1. Hardware Requirements

Our plan is to make our speech processing efficient but due to the nature of signal processing and many AI algorithms running, we need high grade processing power to maintain real-time results. Computing power over i3 processor and 4 GB of RAM will be a minimum hardware specification. For further optimization, we might have to implement the algorithms in parallel architecture for that, we will require GPUs for computation.

1.5.2. Software Requirements

We will be utilizing almost all of the development tools based on open source. Thus this requirement is readily fulfilled. The core packages we will be using are:

- Python IDE
- SciPy
- Pyside-QT
- numPy

1.5.3. Other Requirements

We will need huge corpus of natural language speech and accompanying transcript to train our models. The models we have require both acoustic modelling and language modeling thus we require both the speech data and the language corpus. We will train our language model via web corpus. But for training the acoustic and to train the transcribing module, we need the speech data with the accompanying transcript. However, for the sake of proof of concept the system is designed within limited trained vocabulary

1.6. Feasibility

1.6.1. Operational Feasibility:

Speech Recognition System or Speech to Text systems have been around for long. Their utilities have been proved in various sectors. STT systems have been found to reduce operation cost, help illiterate users, save operation time and much more. The project aims to have the same effect on Nepali Language speaking users in long run. Thus the project seems operationally feasible as end user is likely to save a lot of time and effort by the use of the system.

1.6.2. Technical Feasibility:

Our project mainly requires expertise in Python, Qt, and various algorithm implementations as listed later. We have our team with experience in the above mentioned topics as similar platforms and combinations were used for previous projects. We had also worked on a project called Sambadak: Nepali Text to Speech. The same knowledge can help on this project too.

1.6.3. Schedule Feasibility:

The project needs some complex algorithms and models to work upon. This requires adequate research, which in turn requires a major portion of the allocated time of about nine months. Should time permit, accuracy enhancement and optimization for local language model could be possible enhancements on the project. Thus, for the given time limit, we should be at least able to complete the project with a basic working prototype. Hence the project should be feasible for the given time period.

1.6.4. Cost Feasibility:

The major cost in developing the proposed system is for data. We plan to use archives of news media such as radio or television audio for training the system. Thus, if the same could be obtained for minimal charge under initiation of the department, the expense could greatly be minimized. Also, only free software component will be used for developing the project. Hence, the cost feasibility can be satisfied.

2. LITERATURE REVIEW

2.1. Human Hearing

The human ear is an exceedingly complex organ. To make matters even more difficult, the information from two ears is combined in a perplexing neural network, the human brain. There are still many subtle effects and poorly understood phenomena related to human hearing. We will briefly describe the major components of the ear.

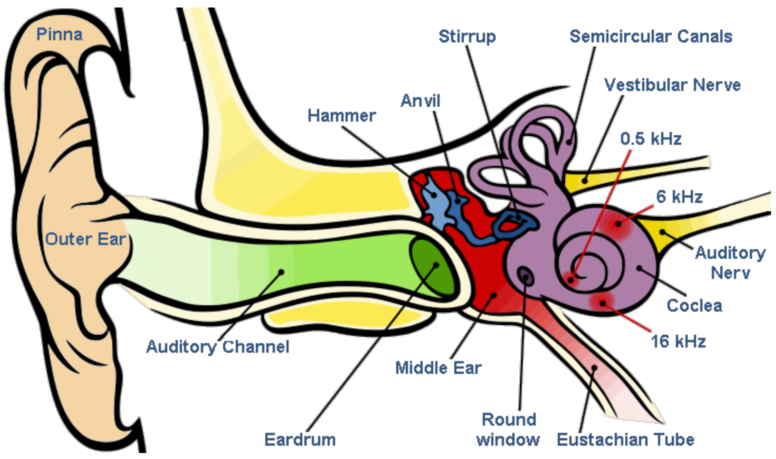


Figure 2.1: Human Auditory System

The ear has three main parts. They are called the outer, middle, and inner ears.

2.1.1. The Outer Ear

The outer ear consists of the organ on the side of our heads that is called pinna. Also included in the outer ear is the ear canal. This is the hollow tube that leads from the pinna into the head. It terminates in the eardrum which is technically known as the tympanic membrane. The purpose of the external ear is to transmit sounds from the outside world into the more internal parts of the auditory system. While one can simply think of the pinna and ear canal as a simple funnel for collecting sounds, in reality they perform some important functions. The pinna has various ridges and folds that act to reflect and absorb certain frequency components of the sound wave. Because the pinna is not circularly symmetric, sounds which come from different directions will have slightly different spectral characteristics. (This means that certain frequencies will be slightly louder or softer depending on the direction they enter the ear.) As a result, sounds which come from above our heads seem slightly different than sounds coming from below. This allows us to localize (pinpoint the direction of) a sound source.

The ear canal a tube about 0.5 cm in diameter extending about 3 cm into the head also plays a role in shaping the spectrum of incoming sounds (emphasizing certain frequencies and attenuating others). It does this in a manner similar to an organ pipe where certain wavelengths tend to reflect back in such a manner so as to cause constructive interference which acts to strengthen the sound. Others frequencies reflect back in a way which causes destructive interference and are thereby weakened. So the net result is that some signal processing already occurs in the outer ear.

Stretched across the end of the ear canal is a thin sheet of tissue called the tympanic membrane or ear drum. Sound waves striking the tympanic membrane cause it to vibrate. It is a small membrane about 1 cm in diameter. Its purpose is to seal off the delicate organs of the inner parts of the auditory system so that foreign matter and bacteria don't enter which could otherwise clog the system and cause harmful infections, as well. However, it is designed to efficiently transmit sound across it.

2.1.2. Middle Ear

The middle ear begins at the inner surface of the eardrum. It is connected to a chain of three small bones called the ossicles. Their names are the malleus (hammer) incus (anvil) and stapes (stirrup) since they resemble those objects in shape. Their purpose is to act as a mechanical transformer. The reason for this is that sound waves are vibrations of air molecules. However, the organ which performs the actual transduction (conversion of acoustic energy into electrochemical impulses) is a fluid filled bony coil called the cochlea.

Because air is much less dense than liquid, and is also more compressible, most of the energy of the sound wave would simply be reflected back into the ear canal. When a sound wave tries to pass from air into liquid, only a small fraction of the sound is transmitted through the interface, while the remainder of the energy is reflected. This is because air has a low mechanical impedance (low acoustic pressure and high particle velocity resulting from low density and high compressibility), while liquid has a high mechanical impedance.

This difference in mechanical impedance results in most of the sound being reflected at an air/liquid interface.

In order to efficiently transmit sound from air into liquid, a lever system is needed to help the move the fluid. The middle bone acts as a sort of pivot for the malleus which is attached to the eardrum, and the stapes which is attached to the cochlea. The two lever arms have different lengths giving a mechanical advantage. See figure 1. In addition, the areas of those surfaces of the bones that are in contact with the eardrum and with the cochlea are unequal, as in a hydraulic lift, giving an additional mechanical advantage. In engineering terms, this process is called impedance transformation. The middle ear is an impedance matching network that increases the fraction of sound energy entering the liquid of the inner ear. Most of the impedance conversion results from the difference in area between the ear drum (receiving sound from the air) and the oval window. The ear drum has an area of about 60 (mm)2, while the oval window has an area of roughly 4 (mm)2. Since pressure is equal to force divided by area, this difference in area increases the sound wave pressure by about 15 times.

2.1.3. Inner Ear

The inner ear refers to the cochlea which is a helically shaped bony structure that resembles a snail in appearance. It is the most complicated part of the auditory system. Basically, it has three parallel fluid filled channels that coil the around the axis of the cochlea. If we were to hypothetically unwind the cochlea it would look something like in the above figure . The last of the middle ear bones, the stapes, acts like a piston, and pushes on the fluid in the first channel, the scala vestibuli, through an opening in the base of the cochlea called the oval window, setting up a pressure wave. The pressure wave propagates in this fluid which is called perilymph toward the end of the cochlea which is called the apex. At the apex, the scala vestibuli connects through an opening called the helicotrema to the third channel which is called the scala tympani. The scala tympani acts as a return path for the pressure wave back towards the base of the cochlea. There is a flexible termination at the basal end of this channel called the round window which bulges in and out with the flow of fluid to allow the wave to flow unimpeded. Otherwise, the incompressible fluid would not have any freedom to cause much motion.

The partition that separates the third channel, the scala tympani from the middle channel, the scala media is called the cochlear partition or the basilar membrane. This membrane

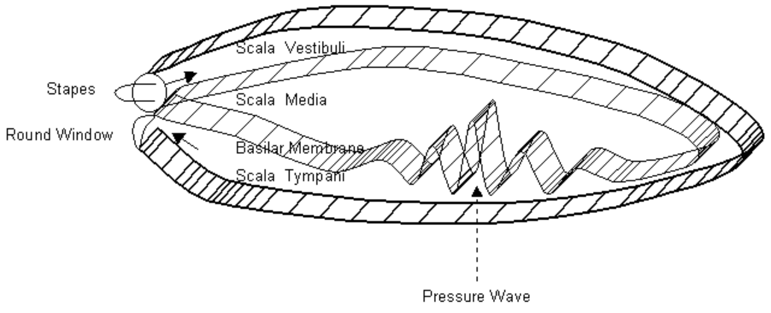


Figure 2.2: Functional Representation of Cochlea

bounces up and down in response to the pressure wave like a toy boat floating in a bathtub in response to a push on the bath water. Located on this membrane is a structure known as the organ of Corti which contains the sound sensing cells. These cells are called hair cells because they have thin bundles of fibers called stereocilia that protrude from their top. As the basilar membrane moves up and down, the stereocilia are pushed back and forth at an angle against a small roof that overhangs the organ of Corti called the tectorial membrane.

The hair cells do not produce action potentials themselves, they release neurotransmitter at synapses with the fibers of the auditory nerve, which produce action potentials. The patterns of oscillations on the basilar membrane are converted to spatiotemporal patterns of firings which transmit information about the sound.

The insides of the hair cells are normally at about a –40 millivolt potential with respect to the normal extracellular (outside the cells) potential of the body. This is called their resting potential. They achieve this by pumping positive ions like sodium (Na+) out of the cell, as do many cells in the body, leaving a net negative charge. But the tops of the hair cells are located in the middle channel, the scala media, which contains a special fluid called endolymph that is potassium (K+) rich and has a +80 millivolt positive charge. This is thought to be generated by special cells in a region on the side of the scala media known as the stria vascularis. The net result of all this is that effectively this creates a battery, as it were, whose positive terminal is connected to the outside of the upper surface of the hair cell, and whose negative terminal is inside the hair cell. When the stereocilia move back and forth, small pressure gated ion channels in the hair cell membrane open which let K+ ions flow into the hair cells from the endolymph. They rush in because the strongly negative potential of the hair cells attracts positive ions. This tends to neutralize some of the negative charge, and brings the potential up towards zero, a process known as depolarization.

On the bottom and sides of the hair cells there are inputs to the auditory nerve which are separated from the hair cell by a small gap called a synapse. When depolarization occurs, special voltage sensitive calcium (Ca 2+) channels open in the hair cell, triggered by the voltage rise and let calcium into the cell. The calcium causes the hair cell to release a quantity of a special chemical called a neurotransmitter that stimulates the auditory nerve to fire. (It does this by forcing the nerve to open up its own ion channels at the synapse which raises the voltage in the nerve fiber. This triggers still other adjacent voltage sensitive channels in the fiber to open which results in a domino effect that causes a wave of depolarization to propagate along the entire nerve.)

2.1.4. Transduction

Contained within the cochlea is the basilar membrane, the supporting structure for about 12,000 sensory cells forming the cochlear nerve. The basilar membrane is stiffest near the oval window, and becomes more flexible toward the opposite end, allowing it to act as a frequency spectrum analyzer. When exposed to a high frequency signal, the basilar membrane resonates where it is stiff, resulting in the excitation of nerve cells close to the oval window. Likewise, low frequency sounds excite nerve cells at the far end of the basilar membrane. This makes specific fibers in the cochlear nerve respond to specific frequencies. This organization is called the place principle, and is preserved throughout the auditory pathway into the brain.

Another information encoding scheme is also used in human hearing, called the volley principle. Nerve cells transmit information by generating brief electrical pulses called action potentials. A nerve cell on the basilar membrane can encode audio information by producing an action potential in response to each cycle of the vibration. For example, a 200 hertz sound wave can be represented by a neuron producing 200 action potentials per second. However, this only works at frequencies below about 500 hertz, the maximum rate that neurons can produce action potentials. The human ear overcomes this problem by allowing several nerve cells to take turns performing this single task. For example, a 3000 hertz tone might be represented by ten nerve cells alternately firing at 300 times per second. This extends the range of the volley principle to about 4 kHz, above which the place principle is exclusively used.

It turns out that there are actually two types of hair cells, the inner hair cells and the outer hair cells. What we have described is the action of the inner hair cells. Outer hair cells are thought to act as amplifiers, not as transducers. They do this by stretching and contracting as the pressure wave passes near by. This pushes the tectorial membrane up and down with more force than would be achieved by the fluid alone. Recall that the tectorial membrane is the roof over the hair cells against (or in close proximity to) which the stereocilia move.

An important property of the basilar membrane is that each region is tuned to a particular frequency. The basal end is tuned to higher frequencies. That means that the point of maximum vibration for higher frequency sounds is at the base. As a high frequency pressure wave enters, it vibrates with a maximum amplitude at a point near the base, and quickly dies out as the wave continues inward. Lower frequency sounds continue on inward until their maximum point of vibration, and quickly die out after that point. Still lower frequency sounds produce maximum vibration at points close to the apex. Therefore, the nerve fibers which are located near the base contain the higher frequency components of the sound, while the fibers located near the apex contain the lower frequencies components of the sound. In this manner, the cochlea acts as a spectrum analyzer, separating different frequencies of sound from each other. It is thought that the outer hair cells contribute to the sharpness of this tuning, with each cell selectively elongating and contracting only in response to its favorite frequency, and not others. The frequency at which a fiber responds best is called its characteristic frequency (CF).

There are many complex circuits and interconnections between the various auditory centers of the central nervous system. In addition, not only does the ear send impulses to the brain, but the brain sends impulses to the ear, as well. This is an example of feedback. It is not fully understood at the present time the exact role of such feedback. One opinion is that the feedback serves as an AGC system to cut down the gain of loud sounds, thus making them easier to hear, and less likely to cause damage. (Overly loud sounds are known to destroy the delicate structures of the auditory system.)

2.1.5. Auditory Signal Processing in the Ear

The basilar membrane of the inner ear spreads out different frequencies: high frequencies produce a large vibration at the end near the middle ear (the "base"), and low frequencies a large vibration at the distant end (the "apex"). Thus the ear performs a sort of frequency analysis, roughly similar to a Fourier transform. However, the nerve pulses delivered to the brain contain both rate-versus-place and fine temporal structure information, so the similarity is not strong.

Briefly, the louder the sound component at a particular frequency, the more frequently the nerve impulses (spikes) are fired in the fiber whose CF (characteristic frequency) is at that particular frequency. Nerve impulses are thought not to vary in amplitude. They are rather like a digital signal. Either a 1 or 0 is transmitted, but nothing in between. The amplitude of the sound is encoded in the rate of firing. The frequency of a sound is encoded by which particular fiber is responding (known as place coding). However, the auditory nerve has been found to have another interesting property. For low frequency signals, the firing pattern of a fiber is synchronized to the sound waveform. This means that during certain portions of the waveform there is a high probability of firing a spike, and during others, a low probability. This is known as phase locking. The stronger the component, the better it phase locks. The net result is that by plotting the firing pattern using a statistical method called a period histogram, one can observe the waveshape of the sound. So in addition to rate and place coding, there is synchrony coding, as well. It is not fully understood how and why all these coding methods are used.

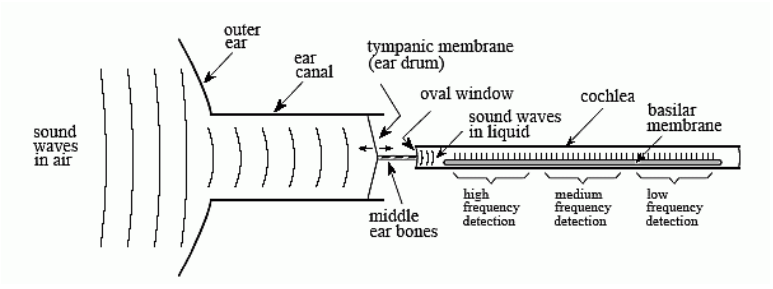


Figure 2.3: Functional diagram of the human ear

One final fascinating phenomenon is the interaction of the two ears in interpreting sound. A big advantage to having two ears is the ability to accurately localize sound. There are two cues that the brain uses in doing this. The first is, as we mentioned earlier, the fact that the time of arrival of sounds in the two ears is slightly different. The closer ear receives the sound slightly earlier. The brain is sensitive to differences in time of arrival of as small as 10 microseconds, and can use this to pinpoint the location of the sound. The second cue is the fact that sounds arrive at slightly different amplitudes in the two ears. This is because the head causes an acoustic shadow which attenuates a sound coming from the opposite direction. By comparing the amplitude of the sounds in the two ears, the location of the source can be identified.

For indoor localization, Sounds bounce numerous times off walls, ceilings, floors and other objects which would totally confuse the brain. It turns out that there is a precedence effect by which the brain only pays attention to the first wavefront that reaches it. All the subsequent echoes are ignored for the purpose of localization.

Watts/cm^2	Decibels SPL	Example Sound
10^{-2}	140 dB	Pain
10^{-3}	130 dB	
10^{-4}	120 dB	Discomfort
10^{-5}	110 dB	Jack hammers and rock concerts
10^{-6}	100 dB	
10^{-7}	90 dB	OSHA limit for industrial noise
10^{-8}	80 dB	
10^{-9}	70 dB	
10^{-10}	60 dB	Normal Conversation
10^{-11}	50 dB	
10^{-12}	40 dB	Weakest audible at 100 hertz
10^{-13}	30 dB	
10^{-14}	20 dB	Weakest audible at 10kHz
10^{-15}	10 dB	
10^{-16}	0 dB	Weakest audible at 3 kHz
10^{-17}	-10 dB	
10^{-18}	-20 dB	

Table 2.1: Human Hearing Loudness Scale

Table shows the relationship between sound intensity and perceived loudness. It is common to express sound intensity on a logarithmic scale, called decibel SPL (Sound Power Level). On this scale, 0 dB SPL is a sound wave power of 10-16 watts/cm2, about the weakest sound detectable by the human ear. Normal speech is at about 60 dB SPL, while painful damage to the ear occurs at about 140 dB SPL. The difference between the loudest and faintest sounds that humans can hear is about 120 dB, a range of one-million in amplitude. This means that the ear can hear sounds whose strength lies anywhere within a range of over 12 orders of magnitude. Listeners can detect a change in loudness when the signal is altered by about 1 dB (a 12% change in amplitude). In other words, there are only about 120 levels of loudness that can be perceived from the faintest whisper to the loudest thunder. when listening to very weak sounds, the ear drum vibrates less than the diameter of a single molecule. This is accomplished by means of an automatic gain control system (AGC) which attenuates the response to louder sounds.

The perception of loudness relates roughly to the sound power to an exponent of 1/3. For example, if you increase the sound power by a factor of ten, listeners will report that the loudness has increased by a factor of about two (101/3 $\approx$ 2). This is a major problem for eliminating undesirable environmental sounds, for instance, the beefed-up stereo in the next door apartment. Suppose you diligently cover 99% of your wall with a perfect soundproof material, missing only 1% of the surface area due to doors, corners, vents, etc. Even though the sound power has been reduced to only 1% of its former value, the perceived loudness has only dropped to about 0.011/3 $\approx$ 0.2, or 20%.

The range of human hearing is generally considered to be 20 Hz to 20 kHz, but it is far more sensitive to sounds between 1 kHz and 4 kHz. For example, listeners can detect sounds as low as 0 dB SPL at 3 kHz, but require 40 dB SPL at 100 hertz (an amplitude increase of 100). Listeners can tell that two tones are different if their frequencies differ by more than about 0.3% at 3 kHz. This increases to 3% at 100 hertz. For comparison, adjacent keys on a piano differ by about 6% in frequency.

Audiograms in humans are produced using an audiometer, which presents different frequencies to the subject, usually over calibrated headphones, at specified levels. The levels are weighted with frequency relative to a standard graph known as the minimum audibility curve, which is intended to represent "normal" hearing. The primary advantage of having two ears is the ability to identify the direction of the sound. Human listeners can detect the

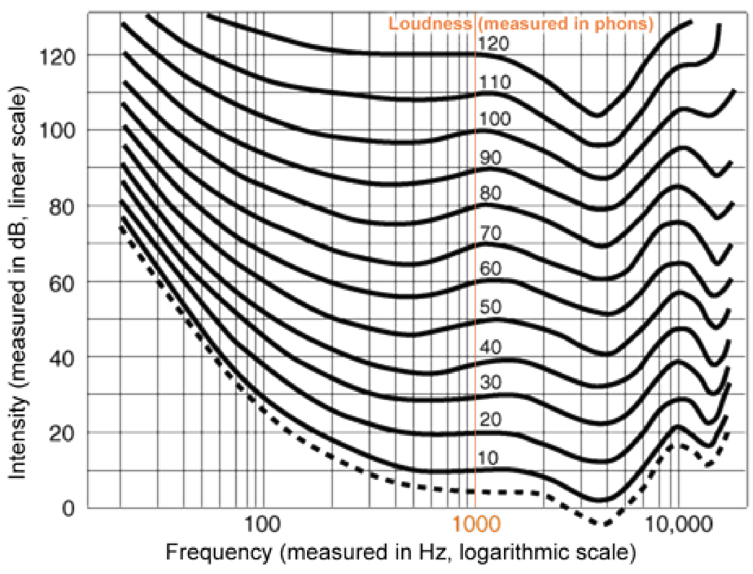


Figure 2.4: Audiogram for perceived loudness

difference between two sound sources that are placed as little as three degrees apart, about the width of a person at 10 meters. This directional information is obtained in two separate ways. First, frequencies above about 1 kHz are strongly shadowed by the head. In other words, the ear nearest the sound receives a stronger signal than the ear on the opposite side of the head. The second clue to directionality is that the ear on the far side of the head hears the sound slightly later than the near ear, due to its greater distance from the source. Based on a typical head size (about 22 cm) and the speed of sound (about 340 meters per second), an angular discrimination of three degrees requires a timing precision of about 30 microseconds. Since this timing requires the volley principle, this clue to directionality is predominately used for sounds less than about 1 kHz.

Both these sources of directional information are greatly aided by the ability to turn the head and observe the change in the signals. An interesting sensation occurs when a listener is presented with exactly the same sounds to both ears, such as listening to monaural sound through headphones. The brain concludes that the sound is coming from the center of the listener's head!

While human hearing can determine the direction a sound is from, it does poorly in identifying the distance to the sound source. This is because there are few clues available in a sound wave that can provide this information. Human hearing weakly perceives that high frequency sounds are nearby, while low frequency sounds are distant. This is because sound waves dissipate their higher frequencies as they propagate long distances. Echo content is another weak clue to distance, providing a perception of the room size. For example, sounds in a large auditorium will contain echoes at about 100 millisecond intervals, while 10 milliseconds is typical for a small office. Some species have solved this ranging problem by using active sonar. For example, bats and dolphins produce clicks and squeaks that reflect from nearby objects. By measuring the interval between transmission and echo, these animals can locate objects with about 1 cm resolution. Experiments have shown that some humans, particularly the blind, can also use active echo localization to a small extent.

2.1.6. Limiting Physics

The performance of the ear at the low end of its dynamic range is limited by internal noise. All signal processing systems have some internal sources of noise due to random fluctuations in various parameters or components of the system. If a signal is too weak, the auditory system can't separate it from these random variations. The high end of the dynamic range is limited by the physical ability of its components to withstand large forces due to the intense vibrations caused by loud sounds.

The frequency response and temporal (time-related) characteristics of the auditory system are limited by the speed at which various steps of the transduction process can occur. There are various mechanical, electrical and chemical bottlenecks that can potentially slow things. Different species have different limits, and can hear much higher pitches than humans can. This is due to variation in the sizes and shapes of various structures in the auditory system among different species.

2.2. Speech Recognition System Overview

The design of Nepali Speech to Text is based on modular architecture. This allows flexibility to design the processing pathway for audio data according to our requirements. This architecture is suitable for testing the performance of different algorithms when they work in unison.

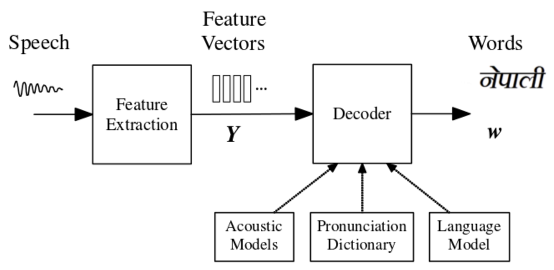


Figure 2.5: Feature Extraction and Decoding

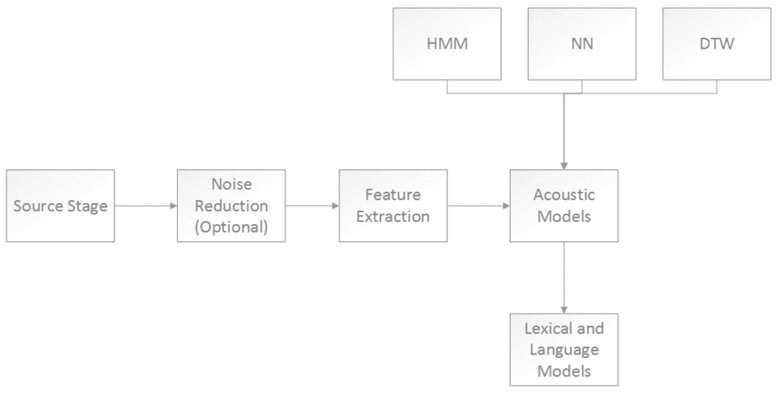


Figure 2.6: System Block Diagram

Here is a brief discussion of each module in the block diagram above.

2.2.1. Source Stage

This stage refers to the source of audio speech data. Most of modern computers have built in microphone which can be used to record audio. Other sources of audio data include audio file of different types like mp3, wav, etc. The software will be able to receive audio data from both audio files and microphone. During training and staging period, the audio file method will be used and during real-time speech to text conversion, microphone data will be used.

2.2.2. Noise Reduction Stage (Optional)

In this stage, the input speech data is filtered to remove external noise. Different methods can be used for noise reduction in real-time audio. We can remove the noise cancellation stage altogether by training the Hidden Markov Models under some noisy conditions that are present at the recognition stage. However, training HMMs at the exact test conditions is practically impossible.

We can use spectral subtraction to preprocess the noisy speech and enhance recognition. Spectral subtraction method is a simple and effective method of noise reduction. In this method, an average signal spectrum and average noise spectrum are estimated in parts of the recording and subtracted from each other, so that average signal-to-noise ratio (SNR) is improved. It is assumed that the signal is distorted by a wide-band, stationary, additive noise, the noise estimate is the same during the analysis and the restoration and the phase is the same in the original and restored signal.

Another way to deal with noise is to compensate the HMMs trained in clean conditions and to adapt them to represent the corrupted speech. The algorithm is known as Parallel Model Combination (PMC) .

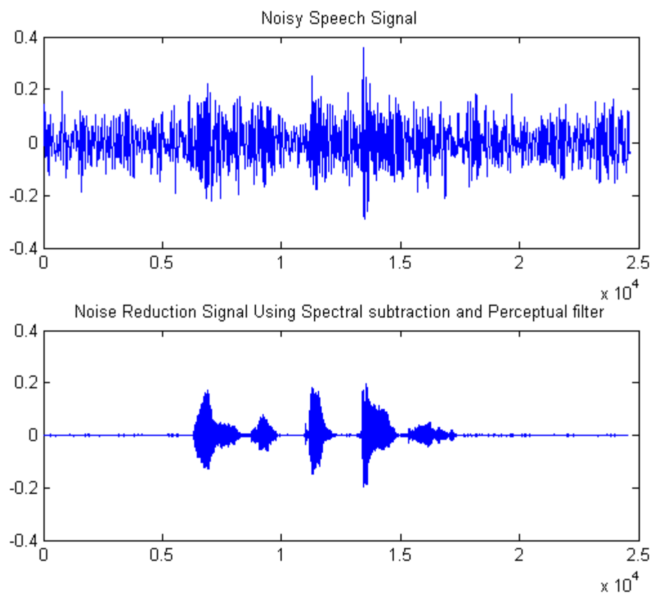


Figure 2.7: Noise Reduction using Spectral Subtraction

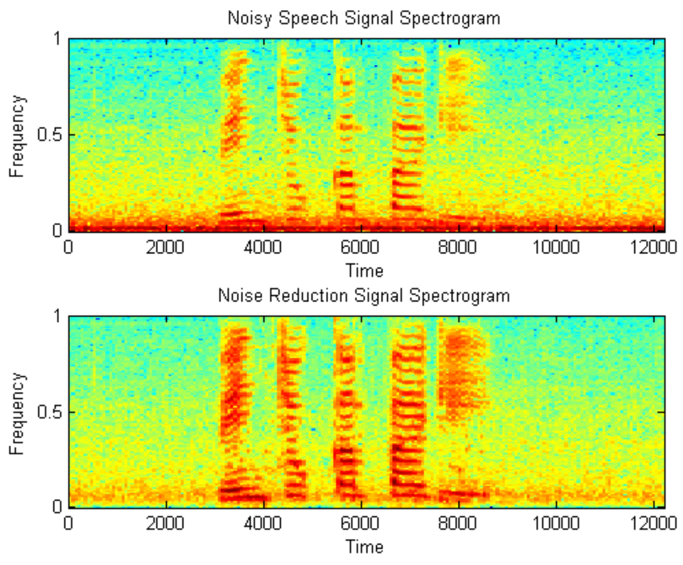


Figure 2.8: Spectral View of Noisy and Noise Reduced Signal

2.2.3. Feature Extraction

In the feature extraction step, the speech waveform with a given sampling rate is processed to produce a new representation as a sequence of vectors containing values of features or parameters. The vectors typically comprise of 10 to 39 parameters, and are usually computed every 10 or 20 milliseconds. These parameters are used in succeeding stages in the estimation that this portion of wave corresponds to particular speech unit already stored in the system's vocabulary. The vocabulary is built via training of the system.

The sampled speech can be analyzed by doing LPC (Linear Predictive Coding), Fourier Transform. LPC tries to predict the present signal representation based on the previous signal representation. LPC analysis gives a set of coefficients, which is the characteristic of the sampled signal. Discrete Fourier Transform converts the sampled signal from time domain to frequency domain. This is important because frequency is one of the important pieces of information required to accurately recognize the sound. Cepstral coefficient, which are another type of coefficients, are found to be more accurate in speech recognition. Cepstral coefficients are the coefficients of the Fourier transform representation of the log magnitude of the spectrum. Although Cepstral coefficients are more robust than LPC coefficients, they require huge resources and time We implement the following for the same:

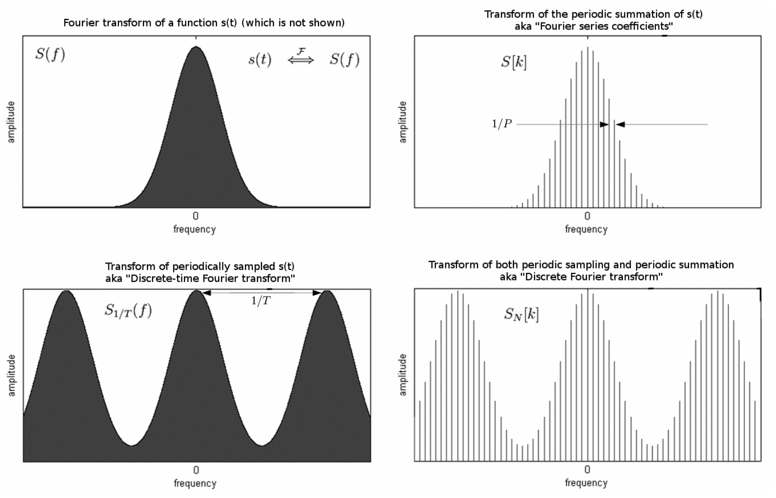


Figure 2.9: Feature Extraction

2.2.4. Acoustic Models

The parameter values extracted from raw speech are used to build acoustic models which approximate the probability that the portion of waveform just analyzed corresponds to a particular phonetic event that occurs in the phone-sized or whole-word reference unit being postulated.

A number of procedures have been developed for acoustic modeling including Dynamic Time Warping (DTW), Hidden Markov Model (HMM), Vector Quantization and Neural Networks. HMM is the dominant recognition paradigm where variations in speech are modeled statistically. Hybrid ANN/HMM systems have also been used for recognition task. In hybrid systems neural networks are used to estimate frame based phone probabilities. These phone probabilities are then converted into the most probable word string using HMM. We will test this stage in both HMM and Neural Network method and choose the best combination.

2.2.5. Lexical and Language Models

Lexical models define the vocabulary for the system whereas language models capture the properties of a language and predict the next word in a speech sequence. The sequence of phone probabilities obtained from acoustic modeling is used for the search of the most likely word by using the constraints imposed by lexical and language models. When speech is produced in a sequence of words, language model or artificial grammars will be used to restrict the combination of words. We will use NGram Markov method to model the language.

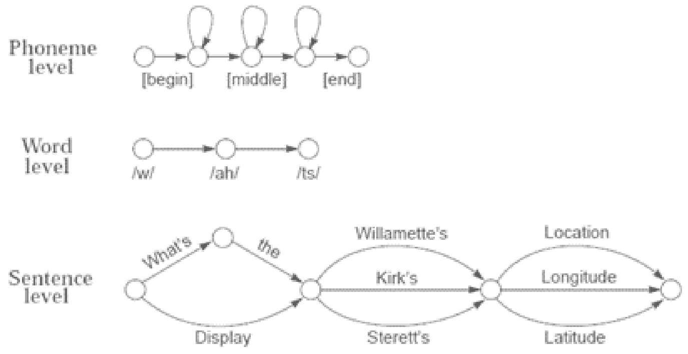


Figure 2.10: Hidden Markov Model Sample

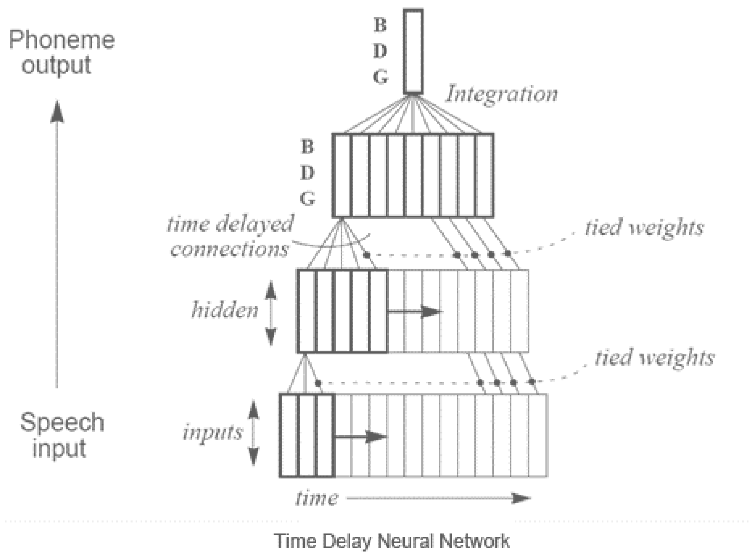


Figure 2.11: Time Delay Neural Network

An n-gram model is a type of probabilistic language model for predicting the next item in such a sequence in the form of a (n - 1)–order Markov model. We will use different sources like the Nepali newspapers and Nepali dictionary to collect data for language models.

2.3. Research Works

2.3.1. Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition

This paper proposes a novel context-dependent (CD) model for large-vocabulary speech recognition (LVSR) that leverages recent advances in using deep belief networks for phone recognition. It describes a pre-trained deep neural network hidden Markov model (DNN-HMM) hybrid architecture that trains the DNN to produce a distribution over senones (tied triphone states) as its output. The deep belief network pre-training algorithm is claimed to be a robust and often helpful way to initialize deep neural networks generatively that can aid in optimization and reduce generalization error. They have described the procedure for applying CD-DNN-HMMs to LVSR, and analyze the effects of various modeling choices on performance. It is said that experiments demonstrate that CD-DNN-HMMs can significantly outperform the conventional context-dependent Gaussian mixture model (GMM)-HMMs, with an absolute sentence accuracy improvement of 5.8% and 9.2% (or relative error reduction of 16.0% and 23.2%) over the CD-GMM-HMMs trained using the minimum phone error rate (MPE) and maximum-likelihood (ML) criteria, respectively.

2.3.2. Speech Recognition using Neural Networks

This thesis examines how artificial neural networks can benefit a large vocabulary, speaker independent, continuous speech recognition system. Currently, most speech recognition systems are based on hidden Markov models (HMMs), a statistical framework that supports both acoustic and temporal modeling. Despite their state-of-the-art performance, HMMs make a number of suboptimal modeling assumptions that limit their potential effectiveness.

Neural networks avoid many of these assumptions, while they can also learn complex functions, generalize effectively, tolerate noise, and support parallelism. While neural networks can readily be applied to acoustic modeling, it is not yet clear how they can be used for temporal modeling. Therefore, the paper explores a class of systems called NN-HMM hybrids, in which neural networks perform acoustic modeling, and HMMs perform temporal modeling. They have argued that a NN-HMM hybrid has several theoretical advantages over a pure HMM system, including better acoustic modeling accuracy, better context sensitivity, more natural discrimination, and a more economical use of parameters. These advantages are confirmed experimentally by a NN-HMM hybrid that they developed, based on context-independent phoneme models that achieved 90.5% word accuracy on the Resource Management database, in contrast to only 86.0% accuracy achieved by a pure HMM under similar conditions.

2.3.3. Language-independent and language-adaptive acoustic Modeling for speech recognition

The research focuses on the question of how to port large vocabulary continuous speech recognition systems in a fast and efficient way as with the distribution of speech technology products all over the world, the portability to new target languages has become a practical concern. They have estimated acoustic models for a new target language using speech data from varied source languages, but only limited data from the target language. For this purpose, they have introduced different methods for multilingual acoustic model combination and a polyphone decision tree specialization procedure. Recognition results using language-dependent, independent and language-adaptive acoustic models are presented and discussed in the framework of our Global Phone project which investigates LVCSR systems.

2.4. Previous Implementations

Speech Recognition is not a new concept and have been known to be around since 1940s. Since its dawn at AT&T Bell Labs, its widespread application possibility has been analyzed and implemented in many forms. In the 1960s, researchers turned their focus towards a series of smaller goals that would aid in developing the larger speech recognition system. As a first step, developers created a device that would use discrete speech, verbal stimuli punctuated by small pauses. However, in the 1970s, continuous speech recognition, which does not require the user to pause between words, began. This technology became functional during the 1980s and is still being developed and refined today.

Most of the research in this field has been geared towards the English language. Some of the promising implementations have been discussed as follows:

2.4.1. Dragon Naturally Speaking:

It is a speech recognition software package developed by Dragon Systems of Newton, Massachusetts, and later acquired by Nuance Communications.. NaturallySpeaking uses a minimal user interface. The user is able to dictate and have speech transcribed as written text, have a document synthesized as an audio stream, or issue commands that are recognized as such by the program. This software used Hidden Markov Model and TriGram Model initially when called as Dragon Dictate.

2.4.2. CMU Sphinx:

CMU Sphinx, also called Sphinx in short, is the general term to describe a group of speech recognition systems developed at Carnegie Mellon University. Sphinx is a continuous-speech, speaker-independent recognition system making use of hidden Markov acoustic models (HMMs) and an n-gram statistical language model. It was developed by Kai-Fu Lee. Sphinx featured feasibility of continuous-speech, speaker-independent large-vocabulary recognition, the most recent version of Sphinx is Sphinx-4. Sphinx 4 is a complete re-write of the Sphinx engine with the goal of providing a more flexible framework for research in speech recognition, written entirely in Java.

2.4.3. Microsoft Speech API:

Microsoft Speech API is one of the most used Speech to Text or speech recognition systems built. The Speech Application Programming Interface or SAPI is an API developed by Microsoft to allow the use of speech recognition and speech synthesis within Windows applications. To date, a number of versions of the API have been released, which have shipped either as part of a Speech SDK, or as part of the Windows OS itself. Applications that use SAPI include Microsoft Office, Microsoft Agent and Microsoft Speech Server.

All of the above implementations have been developed for the English Language as the central language. The implementations that are more important to us should be for Nepali language. By researching we came across the following attempt:

2.4.4. Nepali Speech Recognition (2008):

This project focuses on available solution and basic comparison of efficiencies for the same. This project analyzes the existing models for recognition of Nepali speech. Upon finding the shortcomings of the existing models, a new model called Ear Model, has been made, which is the simulation of how human ear and brain work together for speech recognition. This model was found out to provide better accuracy than conventional methods, but for limited trained data for single alphabets. By using these techniques this project has been able to achieve a speech recognition model that can be helpful to all seeking better model of speech recognition of Nepali language. For this project, a small corpus of Nepali text had also been made.

2.5. Variations from Previous Projects

As seen above, there has been some attempts in the similar topic as we want to explore. However, our project varies with the ones listed above in some critical aspects as listed below in the following points:

1. The Nepali Speech Recognition project focused on utilizing Ear Model (Model mimicking the natural human hearing and processing) for automated nepali speech recognition. However, we plan to use existing and tested algorithms such as Hidden Markov Model with possible modifications.

2. The Nepali Speech Recognition project focused on mainly comparing the available alternatives. The project had limited training and data set. Also, the output was tested using single character speech. We plan to implement extensive training data. Also, we expect the output to be valid for word recognition also.

3. The Nepali Speech Recognition project is mainly algorithmic comparison than actual implementation. We plant to make the system more implementation oriented.

4. We propose to use both models HMM (Hidden Markov Model) and Neural Network model to achieve the proposed output. We plan to use HMM as main analysis engine and Neural Network, in case the minimum threshold probability has not been reached unlike previously attempted projects.

2.6. Variation with Speaker Recognition System:

Speaker recognition, the process of automatically recognizing who is speaking on the basis of individual information included in speech waves, is an important branch of speech processing. This technique makes it possible to use the speaker's voice to identify and verify the user and could be used to control access to various systems. Speaker recognition can be classified into identification and verification. Speaker identification is the process of determining which registered speaker provides a given utterance. Speaker verification, on the other hand, is the process of accepting or rejecting the identity claim of a speaker. However, Speech Recognition refers to obtaining actual text representation of the spoken audio, rather than who spoke it.

Speech features are classified as either low-level or high-level characteristics. High-level speech features are associated with syntax, dialect, and the overall meaning of a spoken message. In contrast, low-level features such as pitch and phonemic spectra are associated much more with the physiology of the human vocal tract. Unlike Automated Speech Recognition System or Speech to Text Systems, it is these low-level features that are used in case of Speaker Recognition systems. In the system, for automatic speaker recognition, these features are generally extracted using Mel-Frequency Cepstral Coefficients (MFCCs). However, STT or ASR, use high level features to obtain the actual spoken text.

Hence, in this way, Speaker Recognition System is different from speech recognition system.

3. METHODOLOGY

3.1. Implementation Details

3.1.1. Input Module

The input module refers to the module that is responsible to receive input from the user. The input in our case is raw speech from the user. The input module as been designed with an intention to ease the recording process. There are two methods for speech recording implemented:

1. Simple Recorder

   The simple recorder is a traditional type speech recording system. This recording involves direct use of the computer hardware microphone to retrieve the signal data.

   Algorithm:

   (a) Initialize the microphone

   (b) Initalize channel number

   (c) Read value from default microphone

   (d) Store chunks temporarily

   (e) If not stopped by the user repeat above two steps

   (f) Store the output to file

2. Smart Recorder

   The smart recorder is a new approach to sound recording, that continuously records as long as the speaker is speaking.

   Algorithm:

   (a) Initialize the microphone

   (b) Sample initial 30ms of audio

   (c) Initialize number of channels

   (d) Read value from default microphone

(e) If audio chunk has audio magnitude less than calculated threshold for duration>duration Specified

Stop and store the output to file

else Store chunk temporarily

3.1.2. Voice Activity Detection/Silence Detection:

Determining the beginning and the termination of speech in the presence of background noise is a complicated problem. The major requirement is with labeling sections of speech samples based on whether they are voiced or unvoiced region of speech. The voiced region is the region that we are concerned with, i.e. it is the frame containing the audio spoken to be recognized by the end user. The labeling is done using calculations over the speech samples; zero crossing and short-term energy functions. The short-term energy and zero crossing rate of speech have been extensively used to detect the endpoints or non-voiced/voiced classification of an utterance. For automatic speech recognition, endpoint detection is required to isolate the speech of interest so as to be able to create a speech pattern or template. The process of separating the speech segments of an utterance from the background, i.e., the non speech segments obtained during the recording process, is called endpoint detection. In isolated word recognition systems or Voice Activity Detection problems, accurate detection of the endpoints of a spoken word is important for two reasons, namely: Reliable word recognition is critically dependent on accurate endpoint detection and the computation for processing the speech is less, when the endpoints are accurately located.

The final expected result is to classify as:

- **Unvoiced:**
  Vocal cords are not vibrating, resulting in a periodic or aperiodic speech waveform.
- **Voiced:**
  Vocal cords are tensed and vibrating periodically, resulting in speech waveform that is quasi-periodic Quasi-periodic means that the speech waveform can be seen as periodic over a short-time period (5-100 ms) during which it is stationary

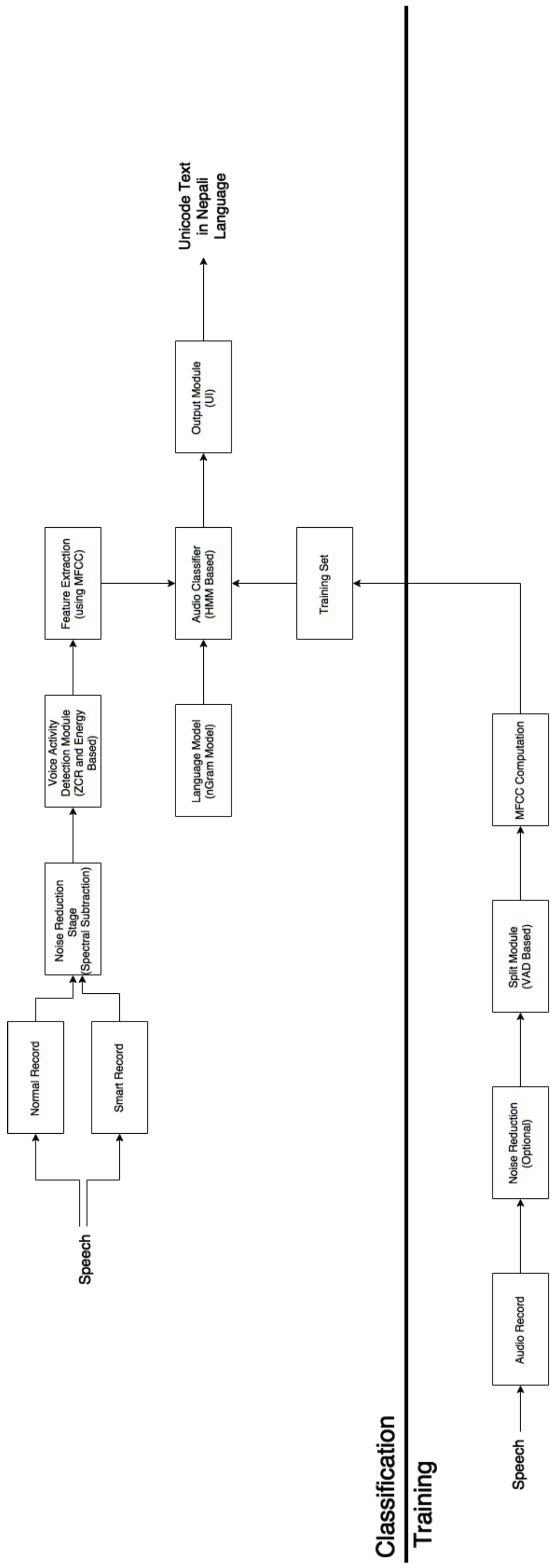

Classification
Training
Speech
Normal Record
Smart Record
Noise Reduction Stage (Spectral Subtraction)
Voice Activity Detection Module (ZCR and Energy Based)
Feature Extraction (using MFCC)
Language Model (nGram Model)
Audio Classifier (HMM Based)
Output Module (UI)
Unicode Text in Nepali Language
Training Set
Speech
Audio Record
Noise Reduction (Optional)
Split Module (VAD Based)
MFCC Computation

Figure 3.1: Implementation Block Diagram

Important Terminologies

1. **Short Term Energy(STE):**

   Short-term energy is the principal and most natural feature that has been used. This is especially to distinguish between voiced sounds and unvoiced sounds or silence. Voiced speech has most of its energy collected in the lower frequencies, whereas most energy of the unvoiced speech is found in the higher frequencies. Different energies used for signal analysis in various forms as per the requirement. As shown below, equation 1 represents Logarithmic Short-Term Energy, equation 2 represents the squared short-Term Energy and equation 3 represents Absolute Short-Term Energy.

$$E_{log} = \sum_{n=1}^{N} log[s(n)^2] \text{.......................}(1)$$

$$E_{sqr} = \sum_{n=1}^{N} [s(n)^2] \text{.......................}(2)$$

$$E_{sqr} = \sum_{n=1}^{N} |s(n)^2| \text{.......................}(3)$$

For simplicity the frame processing used in speech recognition, as stated above. Instead of calculating this parameter with respect to sample, calculations are done on frame basis. According to this approach the, equation is the required estimate.

$$E_{s1}(m) = \sum_{n=m+1-L}^{M} s(n)^2 \text{.......................}(4)$$

The short term energy estimate will increase when speech is present in signal s(n). This is the also case with short term power estimate as follows, which gives a better normalized view:

$$P_{s1}(m) = \frac{1}{L} \sum_{n=m+1-L}^{M} s(n)^2 \text{.......................}(5)$$

2. **Zero Crossing Rate:**

   The number of zero-crossings is also a useful temporal feature in speech analysis. It refers to the number of times speech samples change sign in a given frame. The rate at which zero crossings occur is a simple measure of the frequency content of a narrowband signal. A zero crossing is said to occur if successive samples have different algebraic signs. The rate at which zero crossings occur is a simple measure of the frequency content of a signal. Zero-crossing rate is a measure of number of times in a given time interval/frame that the amplitude of the speech signals passes through a value of zero. The number of zero-crossings, which is also a useful temporal feature in speech analysis, refers to the number of times speech samples change sign in a given frame. This short term zero crossing rates tend to be larger during unvoiced regions. The multiplication by a factor of 1/L is intended to take the average value of the zero-crossing measure.

   $$Z_{s1}(m) = \frac{1}{L}\sum_{n=M-L+1}^{M} \left|\frac{sgn(s(n)) - sgn(s(n-1))}{2}\right| \ldots\ldots\ldots\ldots\ldots\ldots\ldots(6)$$

   where sgn(n)=+1 for s(n) $\geq$ 0 else -1

Algorithm:

The algorithm to be discussed needs some triggers for making decision about where the utterance begins and where to end .To create a trigger one need some information about the background noise. This is done by using following procedure:

1. Assuming that first ten blocks are background noise. With this assumption the mean(μ_w) variance(δ_w) for the measures are calculated. To make a more comfortable approach, the following following function is used as a combination measure:

   $$W_{s1}(m) = P_{s1}(m)(1 - Z_{s1}(m))S_c \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(7)$$

   Where Sc is the scale factor for avoiding small values, in typical application it is 1000 or more.

2. The trigger can then be obtained as,

$$t_w = \mu_w + \alpha\delta_w \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (8)$$

3. The α term is constant that have to be fine tuned according to the characteristics of the signal. The value of 0.92-0.95 seemed to have good effect. But it may vary according to the requirement.

4. The voice activation detection function is then given as follows:

$$VAD(m) = \begin{cases} 1, \mathrm{W}_{s1} \geq t_w \\ 0, \text{otherwise} \end{cases}$$

5. Computing and comparing for each frame gives the desired result.

3.1.3. Noise Reduction Using Spectral Subtraction

The background noise is the most common factor degrading the quality and intelligibility of speech in recordings. The noise reduction module intends to lower the noise level without affecting the speech signal quality. This module is based on the spectral subtraction performed independently in the frequency bands corresponding to the auditory critical bands.

The spectral subtraction method is a method of noise reduction. In this method, an average signal spectrum and average noise spectrum are estimated in parts of the recording and subtracted from each other, so that average signal-to-noise ratio (SNR) is improved. It is assumed that the signal is distorted by a wide-band, stationary, additive noise, the noise estimate is the same during the analysis and the restoration and the phase is the same in the original and restored signal.

The noisy signal y(m) is a sum of the desired signal x(m) and the noise n(m):

$$y(m) = x(m) + n(m)$$

In frequency domain,

$$Y(j\omega) = X(j\omega) + N(j\omega)$$

where $Y(j\omega)$,$X(j\omega)$,$N(j\omega)$ are fourier transforms of y(m), x(m) and n(m).

The statistic parameters of the noise are not known, thus the noise and the speech signal are replaced by their estimates:

$$\hat{X}(j\omega) = Y(j\omega) - \hat{N}(j\omega)$$

The noise spectrum $\hat{N}(j\omega)$ estimate is related to the expected noise spectrum $E[|\hat{N}(j\omega)|]$ which is usually calculated using the time-averaged noise spectrum taken from parts of the recording where only noise is present. The noise estimate is given by:

$$\hat{N}(j\omega) = E[|\hat{N}(j\omega)|] \cong |\overline{N}j\omega)| = \frac{1}{K}\sum_{i=0}^{k-1} |N_i(j\omega)|$$

where $N_i(j\omega)$ is the amplitude spectrum of the i-th of the K frames of noise. Noise estimate in k-th frame may be obtained by filtering the noise using first-order low-pass filter:

$$\hat{N_k}(j\omega) = |\tilde{N_k}(j\omega)| = \lambda_n|\tilde{N_{k-1}}(j\omega)| + (1-\lambda_n)|\tilde{N_k}(j\omega)|$$

where $\hat{N_k}(j\omega)$ is the smoothed noise estimate in i-th frame, n is the filtering coefficient ($0.5 \leq \lambda_n \leq 0.9$, some authors use values $0.8 \leq \lambda_n \leq 0.95$). To obtain the noise estimate, the part of the recording containing only noise that precedes the part containing speech signal should be analysed (the length of the analysed fragment should be at least 300 ms). To achieve this, additional speech detector has to be used.

The spectral subtraction error may be defined as:

$$\varepsilon_{def} = \hat{X}(j\omega) - X(j\omega)$$

This error degrades the signal quality, introducing the distortion known as residual noise or musical noise. The error is a function of expected $E[|\hat{N}(j\omega)|]$ or average $\overline{N}(j\omega)$ noise spectrum estimate:

$$\varepsilon = N(j\omega) - E[|N(j\omega)|] \cong |\overline{N}j\omega)| - E[|N(j\omega)|]$$

Therefore, the longer noise section is used in analysis, the more accurate the noise estimate is.

The signal-to-noise ratio may be defined in frequency domain as SNR a priori (for clean signal) or SNR a posteriori (for noisy signal). SNR in k-th frame is given by:

$$snr_k^{apriori}(\omega) = \frac{[X_k(j\omega)]^2}{[N_k(j\omega)]^2}$$

$$snr_k^{aposteriori}(\omega) = \frac{[Y_k(j\omega)]^2}{[N_k(j\omega)]^2}$$

During the restoration process, the clean signal is not known, hence the SNR a priori value has to be estimated. Using the Gaussian model, optimal SNR in k-th frame may be defined as:

$$\overline{snr}_k^{apriori}(\omega) = (1-\eta)P(snr_k^{aposteriori}(\omega) - 1) + \eta\frac{[\hat{X}_{k-1}(j\omega)]^2}{var(N_{k-1}(j\omega))}$$

$$snr_k^{aposteriori}(\omega) = \frac{[Y_k(j\omega)]^2}{var(N_k(j\omega))}$$

where P(x) is:

$$P(x) = xforx \geq 0and0otherwise$$

$var(N_{k-1}(j\omega))$ is the variance of the noise spectrum in the previous frame, $\hat{X}(j\omega)$ is estimate of the restored signal and is constant $(0.9 < \eta < 0.98)$. The variance is usually replaced by spectral power of noise estimate:

$$\overline{snr}_k^{apriori}(\omega) = (1-\eta)P(snr_k^{aposteriori}(\omega) - 1) + \eta\frac{[\hat{X}_{k-1}(j\omega)]^2}{|N_k(j\omega)^2|}$$

$$snr_k^{aposteriori}(\omega) = \frac{[Y_k(j\omega)]^2}{|N_k(j\omega)^2|}$$

3.1.4. Feature Extraction Using MFCC

In sound processing, the mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency. Mel-frequency cepstral coefficients (MFCCs) are coefficients that collectively make up an MFC. They are derived from a type of cepstral representation of the audio clip (a nonlinear "spectrum-of-a-spectrum"). The difference between the cepstrum and the mel-frequency cepstrum is that in the Mel Frequency Cepstrum, the frequency bands are equally spaced on the mel scale, which approximates the human auditory system's response more closely than the linearly-spaced frequency bands used in the normal cepstrum. This frequency warping can allow for better representation of sound, for example, in audio compression. The first step in any automatic speech recognition system is to extract features i.e. identify the components of the audio signal that are good for identifying the linguistic content and discarding all the other stuff which carries information like background noise, emotion etc.

The main point to understand about speech is that the sounds generated by a human are filtered by the shape of the vocal tract including tongue, teeth etc. This shape determines what sound comes out. If we can determine the shape accurately, this should give us an accurate representation of the phoneme being produced. The shape of the vocal tract manifests itself in the envelope of the short time power spectrum, and the job of MFCCs is to accurately represent this envelope. This page will provide a short tutorial on MFCCs. Mel Frequency Cepstral Coefficents (MFCCs) are a feature widely used in automatic speech and speaker recognition. They were introduced by Davis and Mermelstein in the 1980's, and have been state-of-the-art ever since. Prior to the introduction of MFCCs, Linear Prediction Coefficients (LPCs) and Linear Prediction Cepstral Coefficients (LPCCs) and were the main feature type for automatic speech recognition (ASR).

Basic Overview

The High Level steps for computing MFCC are given below:

1. Frame the signal into short frames, say 20-40 ms (25ms Usual Scenario).
2. For each frame calculate the periodogram estimate of the power spectrum, i.e. basically the modulus squared of the fourier transformed coefficient.
3. Apply the mel filterbank to the power spectra, summing the energy in each filter.
4. Take the logarithm of all filterbank energies, for converting to appropriate scale.
5. Take the DCT of the log filterbank energies.
6. Keep DCT coefficients 2-13, discard the rest the higher DCT coefficients represent fast changes in the filterbank energies and it turns out that these fast changes actually degrade ASR performance, so we get a small improvement by dropping them.

Algorithm Detail

We start with a speech signal, we'll assume sampled at 16kHz.

1. Frame the signal into 20-40 ms frames. 25ms is standard. This means the frame length for a 16kHz signal is 0.025*16000 = 400 samples. Frame step is usually something like 10ms (160 samples), which allows some overlap to the frames. The first 400 sample frame starts at sample 0, the next 400 sample frame starts at sample 160 etc. until the end of the speech file is reached. If the speech file does not divide into an even number of frames, pad it with zeros so that it does. The next steps are applied to every single frame, one set of 12 or specified number MFCC coefficients is extracted for each frame. A short aside on notation: we call our time domain signal s(n). Once it is framed we have si(n) where n ranges over 1-400 (if our frames are 400 samples) and ranges over the number of frames. When we calculate the complex DFT, we get si(k) - where the denotes the frame number corresponding to the time-domain frame. Pi(k) is then the power spectrum of frame i.
2. To take the Discrete Fourier Transform of the frame, perform the following:

$$\sum_{n=1}^{N} s_i(n)h(n)e^{\frac{-j2\pi kn}{N}} for 1 \leq k \leq K$$

where h(n) is an N sample long analysis window (e.g. hamming window as provides the best estimate), and K is the length of the DFT. The periodogram-based power spectral estimate for the speech frame si(n) is then given by:

$$P_i(k) = \frac{1}{N}|S_i(k)|^2$$

This is called the Periodogram estimate of the power spectrum. We take the absolute value of the complex fourier transform, and square the result. We would generally perform a 512 point FFT and keep only the first 257 coefficients.

3. Compute the Mel-spaced filterbank. This is a set of 20-40 (26 is standard) triangular filters that we apply to the periodogram power spectral estimate from the above. The filterbank comes in the form of 26 vectors (assuming the FFT settings fom the above state). Each vector is mostly zero, but is non-zero for a certain section of the spectrum. . To calculate filterbank energies we multiply each filterbank with the power spectrum, then add up the coefficents. Once this is performed we are left with 26 numbers that give us an indication of how much energy was in each filterbank.

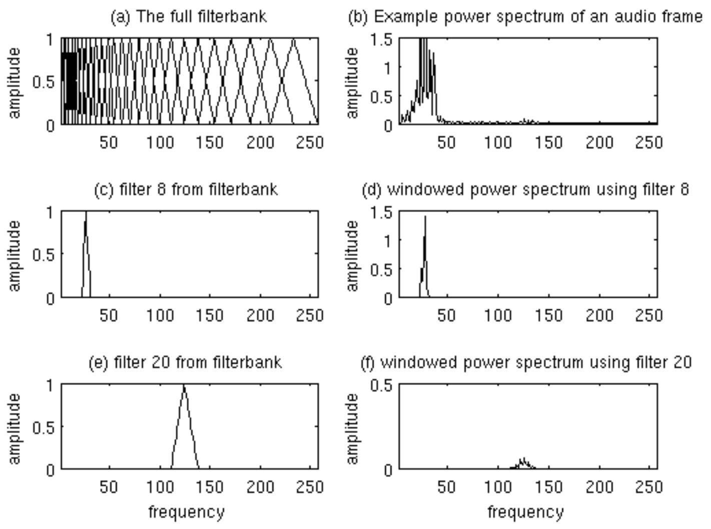


Figure 3.2: Filterbanks for MFCC Computation

4. Take the log of each of the 26 energies from step 3. This leaves us with 26 log filterbank energies.

5. Take the Discrete Cosine Transform (DCT) of the 26 log filterbank energies to give 26 cepstral coefficents. For ASR, only the lower 12-13 of the 26 coefficients are kept.

3.1.5. Language Model

The language model is responsible for correcting the output text of the acoustic model. The sequence of phone probabilities obtained from acoustic modeling is used for the search of the most likely word by using the constraints imposed by lexical and language models. We have used data from various newspaper to build unigram and bigram model for the language model. The data is then utilized by the algorithm to select the best word among the various options by looking at the context of the sentence.

Data Source

We extracted data from different online news portal of Nepal. The data was free to access and provided fairly accurate and complete language model for the test phase. Also, we plan to use news related voice data for training the final system. Hence, data extraction from news related website is a flexible and reliable option. We created scripts to extract data from two major news portal of Nepal : Setopati and NagarikNews.

The news URL of setopati is in the format http://setopati.com/samaj/1234 . We can change the last ID(1234 in this case) to view different news. By visiting different pages and extracting only the news portion, 50MB of text (Unicode) was extracted from setopati. The data contains news related to various fields like politics, sports, etc.

Also, the news URL of nagariknews is in the format http://nagariknews.com/politics/story/1234. Similar to setopati, we can change the ID (1234 in this case) to view different news. 130 MB of raw text (Unicode) was extracted from Nagariknews. The data contains news related to various fields like politics, sports, etc.

Algorithm

We used urllib python library to fetch the destination URL and beautifulsoup library to parse and extract content from the webpages. The algorithm to extract data from the websites is:

1. Start a counter i=1, content = ”
2. For a=1 to 10000
3. Fetch the URL with ID as a
4. Parse the content of the URL to extract only the news portion. Let it be newcontent

5. content = content + newcontent
6. Next a
7. Dump the content in a file

The two text files extracted from the two websites are then merged to a single file. Then, another python program is created that reads the entire content and creates unigram to 5-gram data from the text.

The algorithm to create n-gram data from the text data is as follows:

1. Start
2. Read the contents of the file and also the model number N (like unigram, bigram)
3. Create a list with number of elements N. Initially, all the content of the list is set as '#' to represent "Start of sentence".
4. Create a dictionary of word list and frequency count
5. For each word in the text
6. Append the word to the last of the list.
7. Remove the first item of list if the size of list is greater than N.
8. Add 1 to the dictionary value with the list value as the key.
9. Next word
10. Arrange the dictionary in the ascending order of frequency
11. Dump the content of the dictionary in a file with the word list and frequency count.

We created uni-gram, bi-gram, trigram, 4-gram and 5-gram language model data from the text content.

Spell Correction and Text Prediction

The purpose of the language model is to correct the output of the acoustic model. It will take input from the acoustic model as mispelled words (in most cases). It will then create a list of most probable words in the context using the n-gram language model data. The information is then used to predict the actual word input by the user in the acoustic model.

The first step in the Spell correction and Text prediction process is to get a list of probable correct word for every word generated by the acoustic model. This is done by calculating the edit distance of the correct words from the probably misspelled word. A probable word list (candidate list) for any word is generated by evaluating all the words that fall in the range of certain edit distance from the word. As we are dealing with Nepali words, we can ignore certain aspects of the word like ookar and iikar so that we can compare the words in their simplified form to generate more wide range of words. All the unigram and bigram data are also converted to their bare form.

The edit distance between two words is given by :

$$d_{i0} = \sum_{k=1}^{i} w_{del}(b_k) for 1 \leq i \leq m$$

$$d_{0j} = \sum_{k=1}^{j} w_{ins}(a_k) for 1 \leq j \leq m$$

$$d_{ij} = \begin{cases} \mathrm{d}_{i-1,j-1} for a_j = b_j \\ \min \begin{cases} \mathrm{d}_{i-1,j} + w_{del}(b_i) \\ \mathrm{d}_{i,j-1} + w_{ins}(a_j) \\ \mathrm{d}_{i-1,j-1} + w_{sub}(a_j, b_i) \end{cases} \end{cases}$$

For each bare form of the word, all the other words within edit distance of 1 or 2 are generated and compared to the unigram model. Then, only the words that exist in the unigram model are selected from the word list as a set of possible candidates. The edit distance of the word in non-bare form (actual word comparision) is also considered while evaluating the probability of the candidates.

The most prominent concept of this spell corrector is based upon Baye's Theorem. Mathematically, Bayes's theorem gives the relationship between the probabilities of A and B, P(A) and P(B), and the conditional probabilities of A given B and B given A, P(A|B) and P(B|A). In its most common form, it is:

$$P(A|B) = P(B|A)P(A)/P(B)$$

In our case, the relation has been converted to

$$P(C|W) = P(W|C)/P(C)$$

The unigram probability of a word is given by

$$P(W_i) = \frac{C(i)}{N}$$

Similarly, the bigram probability of a word is given by

$$P(W_i|W_{i-1}) = \frac{C(W_{i-1} - W_i}{C(W_{i-1}}$$

Instead of using only unigram or bigram model, we can combine these two model by interpolating the result. Hence, we can calculate the combined probability as

$$P'(W_i|W_{i-1}) = \lambda * P(W_n|W_{n-1}) + \lambda_2 * P(W_n)$$

where

$$\lambda_1 + \lambda_2 = 1.0$$

The sum of two lambda (interpolation parameters) is always 1.0 so that we will get probability output. We can extend this relation to use bigram, trigram or higher models. Then,

$$\sum_n \lambda_n = 1.0$$

3.1.6. Hidden Markov Models

Modern architectures for Automatic Speech Recognition (ASR) are mostly software architectures generating a sequence of word hypotheses from an acoustic signal. The most popular algorithms implemented in these architectures are based on statistical methods.Hidden Markov Models (HMMs) provide a simple and effective framework for modelling time-varying spectral vector sequences. As a consequence, almost all present day large vocabulary continuous speech recognition (LVCSR) systems are based on HMMs.A vector y_t of acoustic features is computed every 10 to 30 msec.Sequences of vectors of acoustic parameters are treated as observations of acoustic word models used to compute $P(y_1^T|W)$, the probability of observing a sequence y_1^T of vectors when a word sequence W is pronounced.

Structure of a HMM

A hidden Markov model is defined as a pair of stochastic processes (X,Y). The X process is a first order Markov chain, and is not directly observable, while the Y process is a sequence of random variables taking values in the space of acoustic parameters, or observations.

Letting $y\varepsilon Y$ be a variable representing observations and $i, j\varepsilon X$ be variables representing model states, the model can be represented by the following parameters:

$$A \equiv \left\{a_{i,j}|i, j\varepsilon X\right\} \textit{Transition Probabilities}$$

$$B \equiv \left\{b_{i,j}|i, j\varepsilon X\right\} \textit{Output Distributions}$$

$$\Pi = \{\pi_i|i\varepsilon X\} \textit{Initial Probabilities}$$

with the following definitions:

$$a_{i,j} \equiv P(X_t = j|X_{t-1} = i)$$

$$b_{i,j}(y) \equiv P(Y_t = y|X_{t-1} = i, X_t = j)$$

$$\pi_i \equiv p(X_0 = i)$$

For convinience, the HMM is generally represented in a complex notation

$$\lambda = (A, B, \Pi)$$

where the symbols stand for terms as defined above.

The Three Basic Problems for HMMs

Evaluation: Given the observation sequence $O = (o_1, o_2, o_3 ... o_n)$, and an HMM model $\lambda = (A, B, \Pi)$, how do we efficiently compute $P(O|\lambda)$, the probability of the observation sequence, given the model.

Decoding: Given the observation sequence $O = (o_1, o_2, o_3 ... o_n)$, and an HMM model $\lambda = (A, B, \Pi)$, how do we choose a corresponding state sequence $Q = (q_1, q_2, q_3 ... o_n)$ that is optimal in some sense (i.e., best explains the observations)

Learning: How do we adjust the model parameters $\lambda = (A, B, \Pi)$ to maximize $P(O|\lambda)$?

Solution to problem 1 : The Forward Procedure

Given the model λ, calculate $P(O|\lambda)$. One way of doing this is to enumerate all possible combinations. Considering that there are N total observable symbols at each state and we wish to calculate the probability of observation sequence $O = (o_1, o_2, o_3, ... o_T)$, There will be N^T state sequences. And, calculating the probability for each such sequence would require $(2T-1)$ multiplications. So, we would require $(2T-1) * N^T$ multiplications and N^T additions. This is inefficient and a more efficient algorithm is required. Forward algorithm is an efficient algorithm for this. Consider the forward variable

$$\alpha_t(i) = P(o_1, o_2, o_3, ... o_t, q(t) = i|\lambda)$$

that is the probability of the partial observation sequence, $o_1, o_2, ... o_t$ (until time t) and state i at time t, given the model λ. We can solve for $\alpha_t(i)$ inductively as follow.

1. Initialization

$$\alpha_1(i) = \Pi_i b_i(o_1), 1 \leq i \leq N$$

2. Induction

$$\alpha_{t+1}(j) = \left\{ \sum_{i=1}^{N} \alpha_t(i) a(ij) \right\} b_j(i)(o_{t+1}), 1 \leq t \leq T-1; 1 \leq j \leq N$$

3. Termination

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

The Backward Prodedure

In a similar manner, we can consider a backward varialbe $\beta_t(i)$ defined as

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, o_{t+3}, ...o_T, q(t) = i|\lambda)$$

That is, the probability of the partial observation sequence from t+1 to the end, given state i at time t and the model λ. Again we can solve for $\beta_t(i)$ inductively as follows:

1. Initialization

$$\beta_T(i) = 1, 1 \leq i \leq N$$

2. Induction

$$\beta_t(i) = \sum_{j=1}^{N} a(ij)b_j(o_{t+1})\beta_{t+1}(j)$$

$$t = T-1, T-2, ...1, and 1 \leq i \leq N$$

Although only one of the above two procedures need to be applied to solve problem one, but for solving problems two and three generally both of these procedures are applied.

Solution to problem 2 : "Optimal" state sequence

Viterbi algorithm is the most widely used solution to the "optimal state" problem. To find the state sequence $q = (q_1, q_2, ...q_T)$, for the given observation sequence $o = (o_1, o_2, ...o_T)$, We need to define the quantity

$$\delta_t(i) = max(q_1, q_2, ...q_{t-1})P(q_1, q_2, ...q_{t-1}, q_t = i, o_1, o_2, ...o_t|\lambda)$$

This is, $\lambda_t(i)$ is the best source (highest probability) along a single path, at time t, which accounts for the first t observations and ends in state i. By induction, we have

$$\delta_{t+1}(j) = max(i)[\delta_{t+1}(j)a(ij)]b_j(o_{t+1})$$

To actually retrieve the state sequence we need to keep track of the arguments that maximized the above equation, for each t and j. We do this via the array $\Psi_t(j)$. The complete procedure for finding the best state sequence can now be stated as follow.

1. Initialization

$$\delta_t(i) = \Pi_i b_i(o_t), 1 \leq i \leq N$$

$$\psi_1(j) = 0$$

2. Recursion

$$\delta_t(i) = max(1 \leq i \leq N)[\delta_{t-1}(i)a(ij)]b_j(o_t), 1 \leq j \leq N, 1 \leq t \leq T$$

$$\psi_t(j) = argmax(1 \leq i \leq N)[\delta_{t-1}(i)a(ij), 1 \leq j \leq N, 1 \leq t \leq T$$

3. Termination

$$P = max(1 \leq i \leq N)[\delta_T(i)]$$

$$q_T = argmax(1 \leq i \leq N)[\delta_T(i)]$$

4. Backtracking

$$q_t = \psi_{t+1}(q_{t+1}), t = T-1, T-2, ...1$$

The above algorithms are implemented in log scale because of low values of probability. Also, it's worth noting that viterbi algorithm is similar to forward implementation of the forward procedure except some subtle differences.

Solution to problem 3 : Parameter Estimation

The third problem is the most difficult problem of HMMs. The algorithm used for this step is the buam-welch algorithm. Baum-Welch re-estimation uses EM (Expected maximization) to determine the HMM parameters. The algorithm is given below:

1. Define $\xi(i, j)$ as the probability of being in state i at time t, and in state j at time $t+1$.

$$\xi(i, j) = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{P(O|\lambda)}$$

$$= \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}$$

Define $\gamma_t(i)$ as the probability of being in state i at time t, given the observation sequence

$$\gamma_t(i) = \sum_{j=1}^{N} 5(i, j)$$

So, $\sum_{t=1}^{T}\gamma_t(i)$ is the expected number of times state i is visited, and $\sum_{t=1}^{T-1}\xi_t(i, j)$ is the expected number of transitions from state i to state j.

Now, after these calculations are made, the parameters of HMM are updated as

Π_i = expected frequency in state i at time $t = 1$ = $\gamma_1(i)$

a_{ij} = (expected number of transitions from state i to state j) / (expected number of transitions from state i):

$$a_{ij} = \frac{\sum \xi_t(i,j)}{\sum \gamma_t(i)}$$

$b_i(k)$ = (expected number of times in state j and observing symbol k) / (expected number of times in state j):

$$b_j(k) = \frac{\sum_{t,o_t=k} \gamma_t(j)}{\sum_t \gamma(j)}$$

Continuous observation densities in HMMs When introducing HMM, we defined the observations of the HMMs to be discrete symbols from a finite alphabet, and therefore we could use a discrete probability density within each state of this model. These are called discrete HMMs. The problem with this approach is that the observations are often continuous in most of the practical applications. Although it is possible to convert such continuous signal representations into a sequence of discrete symbols via vector quantization codebooks etc, there might be serious degradation associated with such discretization of the continuous signal. Hence, it would be advantageous to be able to use HMMs with continuous observation densities to model continuous signal representations directly.

To use continuous observation density, some restrictions must be placed on the form of the model probability density function (pdf) to ensure that the parameters of the pdf can be re-estimated in a consistent way. The most general representation of the pdf, for which a re-estimation procedure has been formulated, is a finite mixture of the form

$$b_j(o) = \sum_{k=1}^{M} C(jk)N[O,\mu_{jk},U_{jk}], 1 \leq j \leq N$$

Where O is the observation vector being modelled.c_{jk} is the mixture coefficient for the kj^{th} mixture in the state j and N is any log-concave or elliptically symmetric density. The mostly used distribution is the Gaussian distribution. The equations of a multivariate gaussian distribution for independent random variables is where μ_j is the mean vector, and the covariance matrix U_{jk} reduces to σ_{ii}, the product

$$f(X = x|\mu,\Sigma) = \prod_{i=1}^{N} \frac{1}{(2\pi)^{\frac{1}{2}}} exp\left\{ -\frac{(x_i - \mu_i)^2}{2\sigma_{ii}^2} \right\}$$

of the covariance matrix.

3.2. Software Development Methodology and Libraries Used

Nepali Speech to Text is a research oriented project. We have used and will be using GIT version control system for collaboration of the project design and development. We have used python as the main programming language for development. Several libraries like numpy, yahmm, etc have been used during the development with necessity.We have followed PEP 8 coding style for python.

The following lists various libraries used in development of the system:

3.2.1. Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

Python uses whitespace indentation, rather than curly braces or keywords, to delimit blocks; this feature is also termed the off-side rule. An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block.The following lists useful data types in python:

Lists

The list type is a container that holds a number of other objects, in a given order. The list type implements the sequence protocol, and also allows you to add and remove objects from the sequence. To create a list, put a number of expressions in square brackets:

L = []

L = [expression, ...]

This construct is known as a “list display”. Python also supports computed lists, called “list comprehensions”. In its simplest form, a list comprehension has the following syntax: L = [expression for variable in sequence] where the expression is evaluated once, for every item in the sequence.

Dictionary

A dictionary is mutable and is another container type that can store any number of Python objects, including other container types. Dictionaries consist of pairs (called items) of keys and their corresponding values. Python dictionaries are also known as associative arrays or hash tables. The general syntax of a dictionary is as follows: dict = ’Alice’: ’2341’, ’Beth’: ’9102’, ’Cecil’: ’3258’ We can create dictionary in the following way as well: dict1 = ’abc’: 456 ; dict2 = ’abc’: 123, 98.6: 37 ; Defaultdict is a data type that can be imported from the collections module which provides a dictionary-ilke data-type with default value. Usually, a Python dictionary throws a KeyError if you try to get an item with a key that is not currently in the dictionary. The defaultdict in contrast will simply create any items that you try to access (provided of course they do not exist yet). To create such a "default" item, it calls the function object that you pass in the constructor (more precisely, it’s an arbitrary "callable" object, which includes function and type objects).

3.3. NumPy

We used Numpy as a scientific computation library. NumPy is the fundamental package for scientific computing with Python. It contains among other things:

1. A powerful N-dimensional array object

2. Sophisticated (broadcasting) functions
3. Tools for integrating C/C++ and Fortran code
4. Useful linear algebra, Fourier transform, and random number capabilities

In our project, NumPy is used as an efficient multi-dimensional container of generic data. It is a general-purpose array-processing package designed to efficiently manipulate large multi-dimensional arrays of arbitrary records without sacrificing too much speed for small multi-dimensional arrays . NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode compiler/interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents. NumPy seeks to address this problem by providing multidimensional arrays and functions and operators that operate efficiently on arrays. Thus any algorithm that can be expressed primarily as operations on arrays and matrices can run almost as quickly as the equivalent C code. Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted, and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars. The core functionality of NumPy is its "ndarray", for n-dimensional array, data structure. These arrays are strided views on memory. In contrast to Python's built-in list data structure (which, despite the name, is a dynamic array), these arrays are homogeneously typed: all elements of a single array must be of the same type. »> import numpy as np

```
»> x = np.array([1, 2, 3])
»> x array([1, 2, 3])
»> y = np.arange(10)  like Python's range, but returns an array
»> y array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

3.4. PyAudio

PyAudio provides Python bindings for PortAudio, the cross-platform audio I/O library. With PyAudio, we can easily use Python to play and record audio on a variety of platforms, such as GNU/Linux, Microsoft Windows, and Apple Mac OS X. PortAudio is a free, cross-platform, open-source, audio I/O library. It lets us write simple audio programs in 'C' or C++ that will compile and run on many platforms including Windows and Macintosh OS X. The API has been designed with intention to promote the exchange of audio software between developers on different platforms.

PyAudio offers the following features that has been of interest in this project:

1. Multiple file format support (wav, mp3, etc.)
2. Audio Playback facility
3. Non-blocking audio playback facility
4. Audio Record facility
5. Support for writing data to file with specific encoding
6. Reading data into chunks

In the implementation PyAudio has been used for reading the file before framing and recording audio to file.

3.5. YAHMM

YAHMM stands for Yet Another Hidden Markov Model library. This module implements Hidden Markov Models (HMMs) with a compositional, graph- based interface. Models can be constructed node by node and edge by edge, built up from smaller models, loaded from files, baked (into a form that can be used to calculate probabilities efficiently), trained on data, and saved.

Implements the forwards, backwards, forward-backward, and Viterbi algorithms, and training by both Baum-Welch and Viterbi algorithms.

Silent states are accounted for, but loops containing all silent states are prohibited. Tied states are also implemented, and handled appropriately in the training of models.

Features:

1. Build the graph node by node and edge by edge, instead of using matrix format
2. States are not limited to a single distribution type. Not only can different states in the same model have different distributions, but a single state can be an arbitrary weighted mixture of any distributions we like, or just be silent.
3. Normal, Exponential, Uniform Gamma, Inverse-Gamma, Discrete, and Lambda distributions implemented, as well as Gaussian, Uniform, and Triangular Kernel Densities (and of course, Mixtures), and a simple way to define our own distributions.
4. Implements forward, backward, forward-backward, viterbi, all in O(states * edges) time, instead of doing full-graph computations, for significant speedups.
5. Both Baum-Welch and Viterbi training implemented, allowing the possibility of tied-states.
6. Auto-normalization of edge weights to sum to 1
7. Options to simplify the graph structure by merging silent states with a single probability 1 out edge
8. Writing and reading of models to allow for time-intensive training followed by human-readable storage for future use

3.6. PyQT

QT is a cross-platform application framework that is widely used for developing application software that can be run on various software and hardware platforms with little or no change in the underlying codebase, while having the power and speed of native applications. Qt is currently being developed both by the Qt Company, a subsidiary of Digia, and the Qt Project under open-source governance, involving individual developers and firms working to advance Qt. Qt uses standard C++ with extensions including signals and slots that simplifies handling of events, and this helps in development of both GUI and server applications which receive their own set of event information and should process them accordingly.

PyQt combines all the advantages of Qt and Python and is used as it has the following significant features:

1. PyQt brings together the Qt C++ cross-platform application framework and the cross-platform interpreted language Python.

2. PyQT supports Qt classes that employ a signal/slot mechanism for communicating between objects thereby making it easy to create re-usable software components.

3. PyQt is able to generate Python code from Qt Designer.

4. PyQt has easy to use event handling system integrated with the Qt designed file.

5. The .ui file generated from Qt Designer can directly be used.

6. It is also possible to add new GUI controls written in Python to Qt Designer.

4. RESULTS AND VALIDATION

The major expected output of our final project within the above mentioned scope was the transcribing of the human spoken speech signal into the written words in the computer. However, several modules work together to achieve the result. These module contribute to the success and failure of the system. The accuracy of the system is also determined by these modules. Thus, the modules we have developed and whose output can directly be seen are listed below:

4.1. Output From Various Modules

The outputs of various units are listed below:

4.1.1. Output From Voice Activity Detection Module

As seen in the output below (Fig. 4.1), the square waves envelope the spoken text. However, some amount of error has been observed. The error is planned to be dealt with during the verification and validation stage of the project by varying the weightage in various equations. Addition of the noise reduction module is certain to simplify the process.

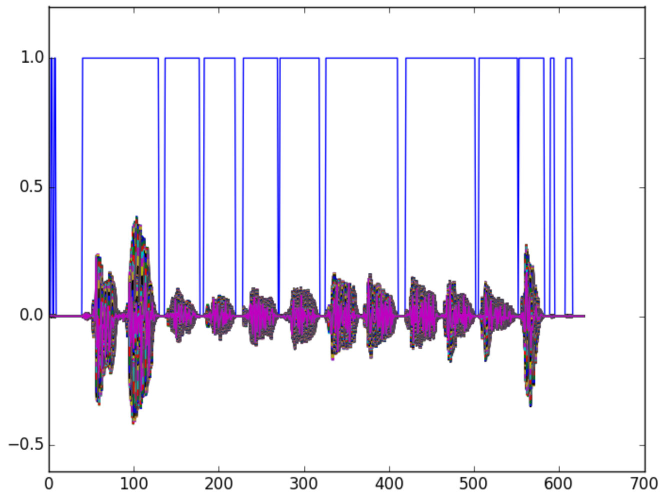
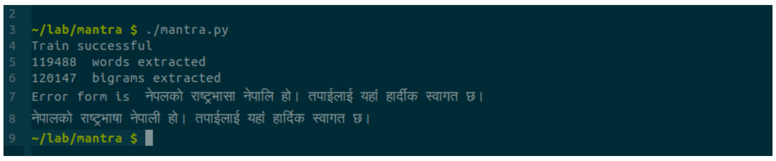


Figure 4.1: Voice Activity Detected Output

4.1.2. Output From the N-Gram Module

As seen in the output below(Fig. 4.2), the erronous form of the input has been correctly identified and corrected based on the probability of occurance obtained from the stored N-Gram data. This module too has certain amount of error which can affect the overall accuracy of the system. However, sincere efforts will be made to minimize the errors as far as possible.



```
~/lab/mantra $ ./mantra.py
Train successful
119488  words extracted
120147  bigrams extracted
Error form is  नेपलको राष्ट्रभासा नेपालि हो। तपाईलाई यहां हार्दीक स्वागत छ।
नेपालको राष्ट्रभाषा नेपाली हो। तपाईलाई यहां हार्दिक स्वागत छ।
~/lab/mantra $ 
```

Figure 4.2: N-Gram Output

4.1.3. Output From the Noise Reduction Module

As seen in the output below, the noisy spectrum form above is cleaned after the spectral subtraction of noise average and shown in the spectrogram below. This boosts the feature extraction process. There still remain smaller artifacts which are short lived notes, which need further more processing to remove. These are called musical noise and after removing those sound is immune to noise to great extent.

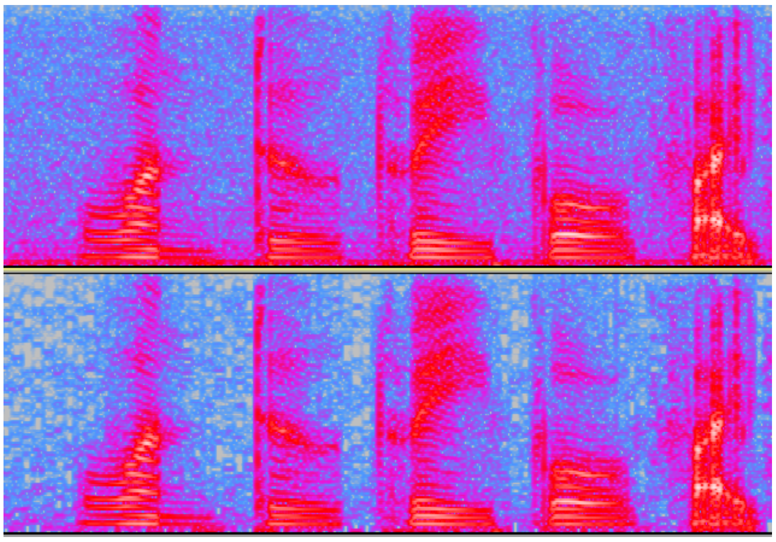


Figure 4.3: Spectral Subtraction Output

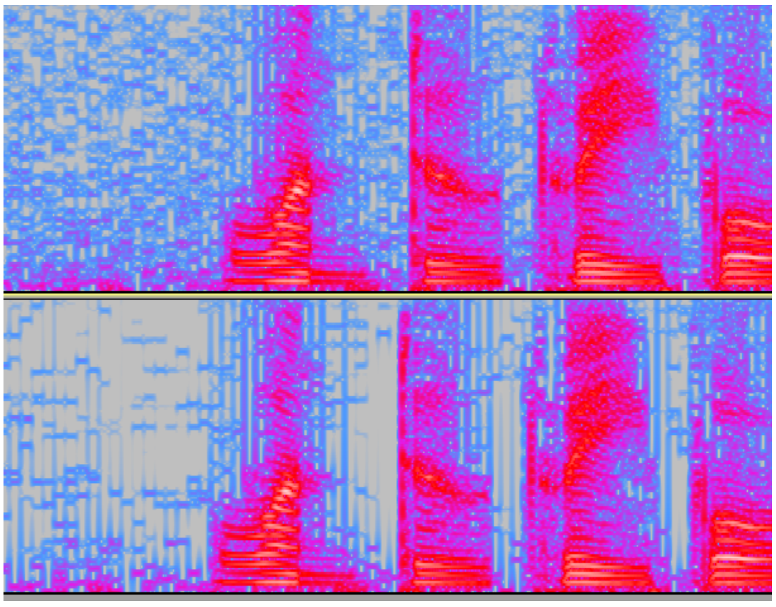
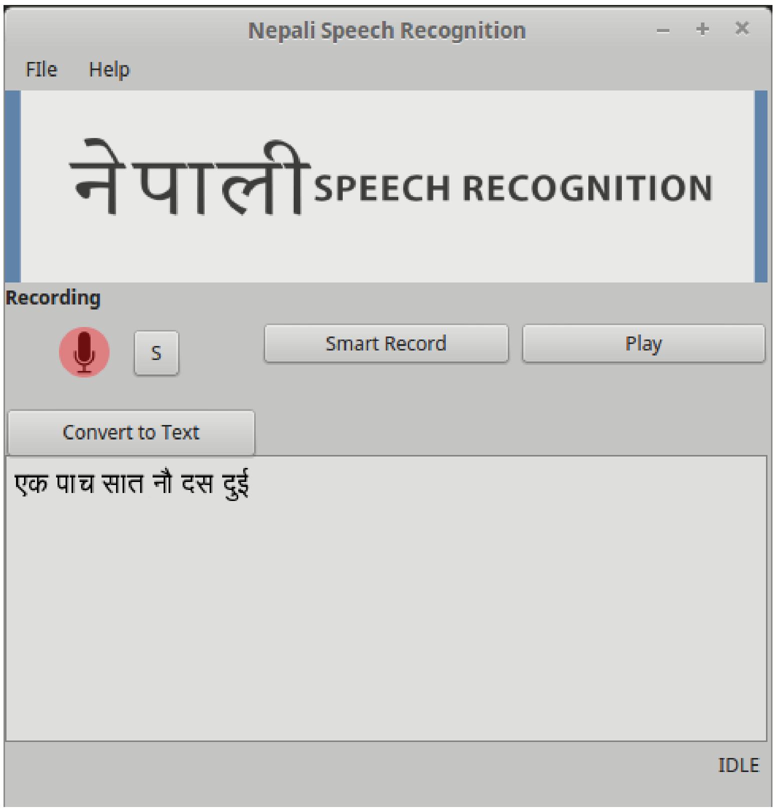


Figure 4.4: After musical noise removal

4.1.4. GUI and the voice recognition

As seen in the output below, the GUI allows for smart recording of the sound by detecting the presence of voice in a continuous way. The recording can be manually be started or stopped too. The recognised words are displayed in the text area below.



एक पाच सात नौ दस दुई

Figure 4.5: GUI and recognised Output

4.2. Validations

Verification and Validation is the process of confirming if we are building the right project and the project right along with the final or overall accuracy of the project itself. For the purpose of calculating overall accuracy we need to find accuracy of individual modules which can be calculated.

The following gives the calculation of individual modules:

4.2.1. Validation for Noise Reduction Module

The noise reduction module is the module responsible for reducing various types of noise such as musical noise and humming sounds from the audio spectrum. This module can affect significantly the overall accuracy of the system.

4.2.2. Approach

For testing the improvement on accuracy of the system, at first the accuracy of the overall system was calculated without the involvement of the module and the accuracy of the overall system was calculated by including the noise reduction module.

4.2.3. Result

The following result was observed:

Without including the noise reduction module

Accuracy obtained = 41%

With Including the noise reduction module

Accuracy obtained = 64%

Analysis:

From the above result it can be seen that the total accuracy of the system has significantly increased compared to the previous case when the noise reduction module was not employed. However, it is to be noted that sometimes, in some cases the error may increase too, due to loss of actual audio information.Also, noise are unpredictable. This can however be tackled using large number of noise profile for training purpose.

4.3. Validation for Audio Split Module

The audio split module is the module in the system that is responsible for splitting a given audio segment from the recorded audio from the user or the training set. The segmentation is a major component as it helps use of large amount of training sets automatically.

4.3.1. Approach

For testing the accuracy of the system, a test set was assigned to be split. After splitting has been completed, the result is checked for

1. Incomplete Word Splitted
2. Empty Split
3. More than one Word in a split

The accuracy of the system is calculated by counting the above error cases for the given test audio.

4.3.2. Result

Expected Number of Words: 100

Obtained Number of Splits: 105

Incomplete Word Split: 10

Empty Splits: 8

More than one word in a Split: 3

4.3.3. Analysis

It was observed that the operation accuracy of the audio split module is not hundred percent as expeceted number of words is not observed. This can be accounted for various facts. The major cause being the variation in noise from the sampling time to the actual speech portion. The error is also caused when the speaker speaks in a comparatively fast manner without introducing silence in between.

The other fact to be considered is that some words inherently contain spaces between them so that error of the first type is introduced. However, the error can be reduced by using a more adaptive type of training method for silence modeling. The error can also be reduced by duration estimation modeling, etc.

4.4. Validation for Language Model

The language model is the module in the system that is responsible for providing probability information for computing or finding out the best possible estimate to a given input string based on the probability of individual word and also the probability of occurance based on its preceeding words.

4.4.1. Approach

For validating the system and calculate the accuracy of the system, various input conditions have been test. The true positive (TP), true negative (TN), false positive (FP) and false negative(FN) parameters are computed for each set of test. Then accuracy for each case is computed and the final overall accuracy is computed.

Table 4.1: Accuracy of Language Model for Various Cases

Test	**Test Case**	**Total(N)**	**TP**	**TN**	**FP**	**FN**	**Accuracy**
I	Single words containing no special characters	50	10	30	1	9	80%
II	Three words containing no special characters	75	40	24	0	11	86.67%
III	More than four words containing no special characters	125	57	31	3	34	70.4%
IV	Single word containing special characters	25	9	2	0	4	84%
V	Two to three words containing special characters	45	20	17	1	7	82.22%
VI	More than four words containing special characters	75	31	27	0	13	77.33%
VII	More than four words not containing special characters but with Numbers	125	57	31	3	34	70.4%
VIII	More than four words containing special characters but with Numbers	75	31	27	0	13	77.33%
						Overall Accuracy	78.54%

4.4.2. Analysis

The accuracy of the language model is satisfactory for a limited domain. It can be seen that the use of numbers and special characters has not significantly affected the round about accuracy of the system. This can be accounted for the fact that the system is designed in such a way that numbers are always regarded as correct entity. Also, the accuracy is not significantly affected by use of special symbols as the system is built in such a way that, correct words are searched for matching the given query. If the crawled words containing the special characters are correct, the system outputs the correct word or sentence.

4.5. Validation of Overall System

This portion covers the overall accuracy of the system integrated as a whole.

4.5.1. Approach

The test was carried out by training the system with a specific set of words. The test cases consist of trained and untrained data. . The true positive (TP), true negative (TN), false positive (FP) and false negative(FN) parameters are computed for each set of test. Then accuracy for each case is computed and the final overall accuracy is computed.

Table 4.2: Accuracy of Overall System Based on Trained and Untrained Data

Set	**Data Type**	**Test Case**	**Total Words(N)**	**Correct Words (C)**	**Incorrect Words (NC)**	**Accuracy**
1	Trained	Single word of varied length	9	8	1	88.88%
2	Trained	Three words of varied length	9	8	1	88.88%
3	Untrained	Single word of varied length	9	6	3	66.67%
4	Untrained	Three words of varied length	9	5	4	55.55%
					Overall Accuracy	74.99%

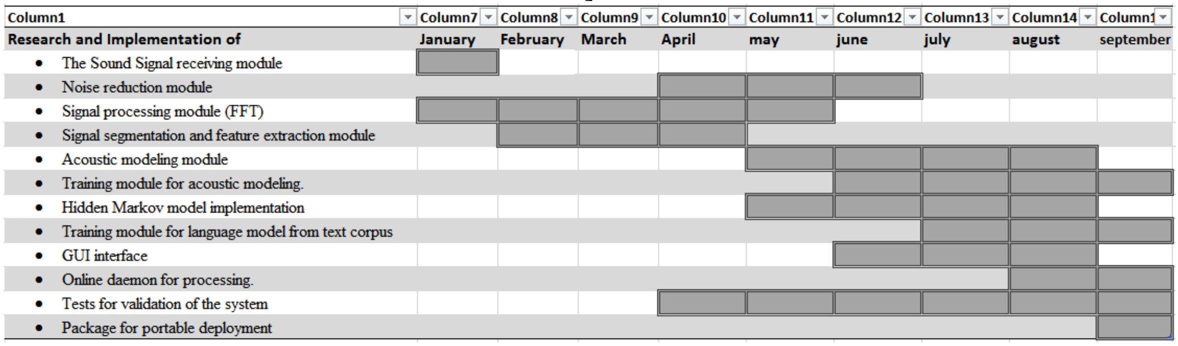
4.5.2. Analysis

The overall accuracy of the system for preliminary case is found to be 74.99%. However, the accuracy varies from cases to cases. The single word accuracy with varied length is found to be 88.88%. The accuracy is not cent percent because varied length scenario creates different word scenario. Also, it is to be noted that, the accuracy of the split module and voice activity module also affects greatly, the accuracy of the system. The clipping operations sometimes is seen to clip significant portion of the spoken audio. The inability of the noise reduction model to operate on non static noise such as horns, coughs, etc. can also cause significant amount of error. Also, it is to be noted that only static MFCC coefficients have been used in the system. By use of Delta and Double Delta coefficients also, the accuracy of the system can be improved.

5. SCHEDULE

As ours is a research based project, the whole process was research oriented and the implementation of the research was done in unison. The schedule of our project is shown in the chart below.

]



Column1	Column7	Column8	Column9	Column10	Column11	Column12	Column13	Column14	Column1
Research and Implementation of	**January**	**February**	**March**	**April**	**may**	**june**	**july**	**august**	september
• The Sound Signal receiving module	▨								
• Noise reduction module				▨	▨	▨			
• Signal processing module (FFT)	▨	▨	▨	▨	▨				
• Signal segmentation and feature extraction module		▨	▨	▨					
• Acoustic modeling module					▨	▨	▨	▨	
• Training module for acoustic modeling.						▨	▨	▨	▨
• Hidden Markov model implementation					▨	▨	▨	▨	
• Training module for language model from text corpus							▨	▨	▨
• GUI interface						▨	▨	▨	
• Online daemon for processing.								▨	▨
• Tests for validation of the system				▨	▨	▨	▨	▨	▨
• Package for portable deployment									▨

Figure 5.1: Project Schedule Gantt Chart

6. CONCLUSION AND FURTHER RECOMMENDATION

6.1. Conclusion

Hence, this project presented various factors and attempts toward utilizing various known speech recognition techniques on Nepali Language. The system presented a preliminary form of implementation with various rooms for improvement. It can be seen that using various techniques and algorithms we were able to obtain overall accuracy of 74.99%. The following represents the possible improvements on the system:

6.2. Further Recommendations

The project was a preliminary attempt on utilizing various known speech recognition techniques on Nepali Language. We cannot claim the system to be complete as a whole, as this has been a completely new paradigm to be explored. There are a lot to be known which can be made concrete only when further research on the Nepali language itself can be carried out. Thus, for better performance of the system the following features are recommended:

1. **Large Amount of Training**: As we know, the accuracy of the whole system largely depends on the amount of training. Thus, using large amount of training data from a large amount of users can greatly improve the accuracy of the system.

2. **Using Deltas and Double Deltas for Feature Extraction**: The feature extraction using MFCC in this report represents static MFCC coefficients. However, speech is dynamic. Thus, using Delta and Double Delta coefficients the accuracy of the system may be increased.

3. **Large Amount of Noise Profile Training**: By training, with a variety of noise profile, the system can have higher accuracy due to better noise reduction.

4. **Speaker Independance**: By using large variety of training data and various normalization techniques, the system can be extended to facilitate speaker independance.

5. **Improvement in the language model**: The language model can be improved for better accuracy. The language model can extensively crawl other data sources for better data representation. Also, modeling according to the Nepali Language model can provide good result

REFERENCES

[1] L. Rabnier and B. Juang. *Fundamentals of Speech Recognition.* Prentice Hall, 1993.

[2] R. Hwouhaybi and Al-Alaoui. *Comparison Of Neural Network for Speaker Recognition.* n.d.

[3] B. Kosko. *Neural Networks and Fuzzy Systeems. A Dynamcal Systems Approach to Machine Intelligence.* Pearson Education, 2008.

[4] A. Marshall. *Artificial Neural Network for Speech Recognition.* 2005.

[5] P. Norvig and S. Russel. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 2009.

[6] C. Prajapati, J. Nyoupane, J.D. Shrestha, and S. Jha. *Nepali Speech Recognition. Kathmandu.* DOECE,IOE, 2008.

[7] T. Schultz and A. Waibe. Language-independent and language-adaptive acoustic Modeling for speech recognition, 2001.

[8] G. E. Dahl, Yu Dong, and Li Den. Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition, 2001.

[9] J. Tebelskis. Speech Recognition using Neural Networks, 1995.

[10] M. Gales and S.Young. Application of Hidden Markov Models in Speech Recognition. `http://mi.eng.cam.ac.uk/~mjfg/mjfg_NOW.pdf`, 2007. [Online].

[11] Wikipedia. Hidden Markov Model. `http://en.wikipedia.org/wiki/Hidden_Markov_model`, 2014. [Online].

[12] Basanta Joshi. Lecture Slides. `http://www.basantajoshi.com.np/courses/SP/`, 2014. [Online].

[13] R. Prasad and A. Sangwan. Comparison of Voice Activity Detection Algorithms for VoIP, 2002.

[14] Y. Huang J. Benesty, M. Sondhi. *Springer Handbook of Speech Processing*. Springer, 2008.

[15] A Acero X. Huang and H. Hon. *Spoken Language Processing*. Preitence Hall, 2001.

[16] A. Downey. *Think Dsp Digital Signal Processing in Python*. Green Tea Press, 2014.

[17] L. Rabiner and R. Schafer. *Digital Processing of Speech Signals*. Prentice Hall, 1978.

[18] P. Wiggers and L.J.M. Rothkrantz. *Automatic Speech Recognition using HMM*. 2003.

[19] Jacob Schreiber. Yet Another Hidden Markov Model repository. `https://github.com/jmschrei/yahmm`, 2014. [Online].

[20] Travis Oliphant. NumPy Numerical Python. `www.numpy.org`. [Online].

[21] N. Dave. Feature Extraction Methods LPC, PLP and MFCC, 2013.